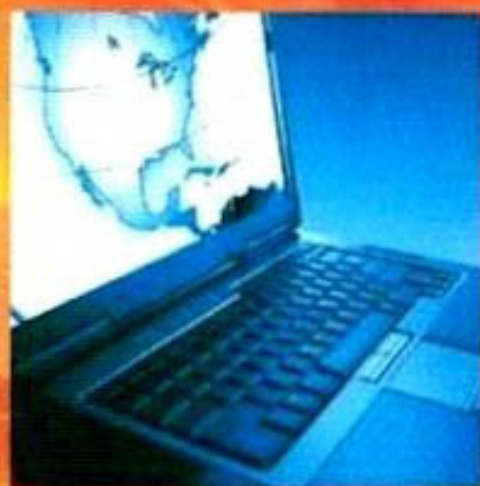


TS. TRẦN ĐOÀN VINH (Chủ biên)
ThS. TRƯƠNG THỊ THU HÀ

HỌC TỐT TIN HỌC 11



NHA XUẤT BẢN
ĐẠI HỌC QUỐC GIA HÀ NỘI

TS. TRẦN DOÃN VINH (Chủ biên)
ThS. TRƯƠNG THỊ THU HÀ

HỌC TỐT TIN HỌC

11

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI

Lời nói đầu

Học tốt Tin học 11 biên soạn theo chương trình Tin học 11 mới của Bộ Giáo dục và Đào tạo, nhằm giúp các em học sinh ôn luyện, đào sâu, nâng cao kiến thức và rèn luyện kỹ năng thực hành Tin học.

Học tốt Tin học 11 có cấu trúc như sau:

- Hệ thống lại các bài trong sách giáo khoa;
- Tương ứng với mỗi bài trong sách giáo khoa có phần bài tập cơ bản và nâng cao nhằm giúp các em học sinh ghi nhớ kiến thức đã học và vận dụng làm để các câu hỏi, bài tập và thực hành.
- Sau phần bài học có phần bài tập thực hành nhằm giúp các em khắc sâu kiến thức đã học, rèn luyện kỹ năng thực hành, tư duy thuật Toán, lập trình, ...

Các câu hỏi và bài tập cơ bản, phần bài tập thực hành đều có hướng dẫn trả lời chi tiết để học sinh tự học và tự kiểm tra kiến thức của mình. Các câu hỏi và bài tập nâng cao chúng tôi không đưa ra lời giải nhằm mục đích phát huy tính tích cực trong học tập của các em.

Chúng tôi hy vọng rằng, quyển sách sẽ là tài liệu tham khảo bổ ích cho các em học sinh, các bậc phụ huynh và các đồng nghiệp. Cuốn sách hẳn còn một số thiếu sót, rất mong nhận được ý kiến đóng góp từ bạn đọc.

Các tác giả

CHƯƠNG I

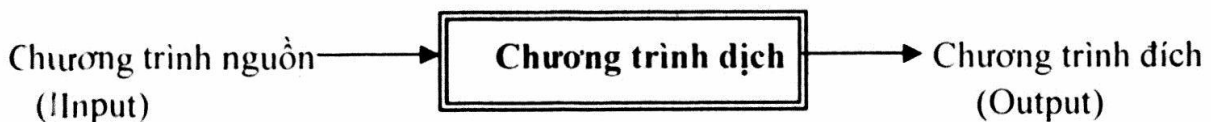
MỘT SỐ KHÁI NIỆM VỀ LẬP TRÌNH VÀ NGÔN NGỮ LẬP TRÌNH

§1. KHÁI NIỆM LẬP TRÌNH VÀ NGÔN NGỮ LẬP TRÌNH

A. TÓM TẮT LÝ THUYẾT

Như chúng ta đã biết **ngôn ngữ lập trình** là ngôn ngữ để viết chương trình, là phương tiện dùng để diễn đạt cho máy tính những việc con người muốn máy thực hiện và nó được chia thành các lớp: *ngôn ngữ máy*, *hợp ngữ* và *ngôn ngữ bậc cao*.

- **Ngôn ngữ lập trình bậc cao** là ngôn ngữ lập trình gần với ngôn ngữ tự nhiên hơn, thuận tiện cho đông đảo người lập trình (không chỉ cho những người lập trình chuyên nghiệp).
- **Lập trình** là sử dụng cấu trúc dữ liệu và các câu lệnh của ngôn ngữ lập trình cụ thể để mô tả dữ liệu và diễn đạt các thao tác của thuật toán, là tạo ra các chương trình giải được các bài toán trên máy tính.
- Chương trình viết bằng ngôn ngữ lập trình bậc cao nói chung không phụ thuộc vào máy, nghĩa là một chương trình có thể thực hiện trên nhiều loại máy. Chương trình viết bằng ngôn ngữ máy có thể được nạp trực tiếp vào bộ nhớ và thực hiện ngay còn chương trình viết bằng ngôn ngữ lập trình bậc cao phải được chuyển đổi thành chương trình trên ngôn ngữ máy mới có thể thực hiện được.
- Chương trình đặc biệt có chức năng chuyển đổi chương trình được viết bằng ngôn ngữ lập trình bậc cao thành chương trình thực hiện trên máy tính cụ thể được gọi là **chương trình dịch**.
- Chương trình dịch nhận đầu vào là chương trình viết bằng ngôn ngữ lập trình bậc cao (chương trình nguồn) và thực hiện chuyển đổi sang ngôn ngữ máy (chương trình đích):



➤ **Ngôn ngữ máy** là ngôn ngữ duy nhất máy tính điện tử có thể trực tiếp hiểu và thực hiện các câu lệnh.

- Chương trình dịch có 2 loại: *thông dịch* và *biên dịch*.

a) Thông dịch (Interpret) được thực hiện bằng cách lặp lại dãy các bước sau:

1. Kiểm tra tính đúng đắn của câu lệnh tiếp theo trong chương trình nguồn:

2. Chuyển đổi câu lệnh đó thành một hay nhiều câu lệnh tương ứng trong ngôn ngữ máy;
3. Thực hiện các câu lệnh vừa chuyển đổi được.

Như vậy, quá trình dịch và thực hiện các câu lệnh là luân phiên. Các chương trình thông dịch lần lượt dịch và thực hiện từng câu lệnh. Nó thích hợp cho môi trường đối thoại giữa người và hệ thống, được ứng dụng cho các ngôn ngữ khai thác hệ quản trị cơ sở dữ liệu, ngôn ngữ đối thoại với hệ điều hành,...

b) **Biên dịch** (compile) được thực hiện qua hai bước:

1. Duyệt, kiểm tra, phát hiện lỗi, kiểm tra tính đúng đắn của các câu lệnh trong chương trình nguồn;
2. Dịch toàn bộ chương trình nguồn thành một chương trình đích có thể thực hiện trên máy và có thể lưu trữ để sử dụng lại khi cần thiết.

Như vậy, trong thông dịch, không có chương trình đích để lưu trữ, trong biên dịch cả chương trình nguồn và chương trình đích đều có thể lưu trữ lại để sử dụng về sau. Nó được ứng dụng vào việc biên soạn, lưu trữ, tìm kiếm, cho biết các kết quả trung gian,... Toàn bộ các dịch vụ trên tạo thành một môi trường làm việc trên một ngôn ngữ lập trình cụ thể. Ví dụ, Turbo Pascal 7.0, Free Pascal 1.2, Visual Pascal 2.1.... trên ngôn ngữ Pascal, Turbo C++, Visual C++,...

§2. CÁC THÀNH PHẦN CỦA NGÔN NGỮ LẬP TRÌNH

A. TÓM TẮT LÝ THUYẾT

1. Các thành phần cơ bản

Mỗi ngôn ngữ lập trình có 3 thành phần cơ bản, đó là: bảng chữ cái, cú pháp và ngữ nghĩa.

a) **Bảng chữ cái** là tập các kí tự được dùng để viết chương trình. Không được phép dùng bất kì kí tự nào ngoài các kí tự quy định trong bảng chữ cái.

Trong Pascal, bảng chữ cái bao gồm các kí tự:

- Các chữ cái thường và các chữ in hoa của bảng chữ cái tiếng Anh:
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
- 10 chữ số Ả Rập: 0 1 2 3 4 5 6 7 8 9
- Các kí tự đặc biệt:

+	-	*	/	=	<	>	[	]	.	,	(dấu phẩy)
;	#	^	\$	@	&	(	)	{	}	:	' (dấu nháy)
dấu cách (mã ASCII 32)								_ (dấu gạch dưới)			

- Bảng chữ cái của các ngôn ngữ lập trình nói chung không khác nhau nhiều.

b) Cú pháp là bộ qui tắc để viết chương trình, mà dựa vào chúng, người lập trình và chương trình dịch biết được tổ hợp nào của các kí tự trong bảng chữ cái là hợp lệ. Nhờ đó, có thể mô tả chính xác thuật toán để máy tính thực hiện.

c) Ngữ nghĩa xác định ý nghĩa thao tác cần phải thực hiện, ứng với tổ hợp kí tự dựa vào ngữ cảnh của nó.

Tóm lại, cú pháp cho biết cách viết chương trình hợp lệ, còn ngữ nghĩa xác định ý nghĩa của các tổ hợp kí tự trong chương trình.

2. Một số khái niệm

a) Tên

- Mọi đối tượng trong chương trình đều phải được đặt tên theo quy tắc của ngôn ngữ lập trình và từng chương trình dịch cụ thể.
- Tên trong Turbo Pascal là một dãy liên tiếp *không quá 127 kí tự bao gồm chữ số, chữ cái hoặc dấu gạch dưới và bắt đầu bằng chữ cái hoặc dấu gạch dưới*.
- Nhiều ngôn ngữ lập trình (Pascal, chẳng hạn), phân biệt ba loại tên, đó là:
 - ◊ Tên dành riêng;
 - ◊ Tên chuẩn;
 - ◊ Tên do người lập trình đặt.

➤ **Tên dành riêng:** Tên được dùng với ý nghĩa xác định, người lập trình không được sử dụng với ý nghĩa khác và chúng còn được gọi là **từ khóa**.

Ví dụ, một số tên dành riêng:

Trong Pascal: program, uses, const, type, var, begin, end.

Trong C++: main, include, if, while, void.

➤ **Tên chuẩn:** Tên dùng với ý nghĩa nào đó, nhưng người lập trình có thể khai báo và dùng chúng với ý nghĩa và mục đích khác. Ý nghĩa của chúng được qui định trong các *thư viện* của ngôn ngữ lập trình.

Ví dụ, tên chuẩn:

Trong Pascal: abs, sqr, sqrt, integer, longint, byte, real, extended, break

Trong C++: cin, cout, getchar

➤ **Tên do người lập trình đặt:** Tên được dùng với ý nghĩa riêng, xác định bằng cách khai báo trước khi sử dụng và chúng không được trùng với tên dành riêng.

Ví dụ, tên do người lập trình đặt: A1, baitap1, bai_thi,...

b) Hằng và biến

❖ **Hằng** là các đại lượng có giá trị không thay đổi trong quá trình thực hiện chương trình.

- > **Hằng số học** là các số nguyên hay số thực (dấu phẩy tĩnh hoặc dấu phẩy động) có dấu hoặc không dấu. Ví dụ, 2, 0, -5, +18, 1.5, 1.0E-6,...

➤ **Hằng logic** là giá trị *đúng* hoặc *sai* tương ứng với *true* hoặc *false*. Ví dụ, hằng logic trong Pascal: TRUE, FALSE.

➤ **Hằng xâu** là chuỗi kí tự trong bảng chữ cái. Khi viết, chuỗi kí tự này được đặt trong dấu nháy (Pascal dùng dấu nháy đơn, còn C++ dùng dấu nháy kép). Ví dụ, hằng xâu trong Pascal: 'hoctottinhoc11', 'ha noi',...trong C++: "TINHOC", "HA NOI",...

❖ **Biến** là đại lượng được đặt tên, dùng để lưu trữ giá trị và giá trị có thể được thay đổi trong quá trình thực hiện chương trình. Các biến dùng trong chương trình đều phải khai báo.

c) Chú thích

Chúng ta có thể đặt các đoạn chú thích trong chương trình nguồn. Chúng giúp cho người đọc chương trình nhận biết ngữ nghĩa của chương trình đó dễ hơn. Nó không ảnh hưởng đến nội dung chương trình nguồn và được chương trình dịch bỏ qua.

Trong Pascal các đoạn chú thích đặt giữa cặp dấu { và } hoặc (* và *), còn trong C++ là đặt các đoạn chú thích giữa cặp dấu /* và */

TÓM TẮT CHƯƠNG I

- Cần có chương trình dịch để chuyển chương trình nguồn thành chương trình đích.
- Có hai loại chương trình dịch: thông dịch và biên dịch.
- Các thành phần của ngôn ngữ lập trình: bảng chữ cái, cú pháp và ngữ nghĩa.
- Mọi đối tượng trong chương trình đều phải được đặt tên:
 - Tên dành riêng: Được dùng với ý nghĩa riêng, không được dùng với ý nghĩa khác.
 - Tên chuẩn: Tên dùng với ý nghĩa nhất định, khi cần dùng với ý nghĩa khác thì phải khai báo.
 - Tên do người lập trình đặt: cần khai báo trước khi sử dụng.
- Hằng: Đại lượng có giá trị không thay đổi trong quá trình thực hiện chương trình.
- Biến: Đại lượng được đặt tên. Giá trị của biến có thể thay đổi trong quá trình thực hiện chương trình.

B. CÂU HỎI VÀ BÀI TẬP

I. BÀI TẬP CƠ BẢN

1. Tại sao người ta phải xây dựng các ngôn ngữ lập trình bậc cao?

2. Chương trình dịch là gì? Tại sao cần phải có chương trình dịch?
3. Biên dịch và thông dịch khác nhau như thế nào?
4. Hãy cho biết các điểm khác nhau giữa tên dành riêng và tên chuẩn.
5. Hãy tự viết ra ba tên đúng theo quy tắc của Pascal và có độ dài khác nhau.
6. Hãy cho biết những biểu diễn nào dưới đây không phải là biểu diễn hằng trong Pascal và chỉ rõ lỗi trong từng trường hợp:

a) 150.0	b) -22	c) 6.23	d) '43'
e) A20	f) 1.06E-15	g) 4+6	h) 'C
i) 'TRUE'			

Hướng dẫn trả lời câu hỏi và bài tập

1. Người ta phải xây dựng các ngôn ngữ lập trình bậc cao, bởi những lí do sau:
 - Ngôn ngữ lập trình bậc cao gần với ngôn ngữ tự nhiên hơn, thuận tiện cho đông đảo người lập trình (không chỉ cho những người lập trình chuyên nghiệp).
 - Ngôn ngữ lập trình bậc cao nói chung không phụ thuộc vào loại máy, cùng một chương trình có thể thực hiện trên nhiều loại máy khác nhau.
 - Chương trình viết bằng ngôn ngữ bậc cao dễ hiểu, dễ hiệu chỉnh và dễ nâng cấp hơn.
 - Ngôn ngữ lập trình bậc cao cho phép làm việc với nhiều kiểu dữ liệu và cách tổ chức dữ liệu đa dạng, thuận tiện cho mô tả thuật toán.
2. *Chương trình dịch* là chương trình đặc biệt, có chức năng chuyển đổi chương trình được viết trên ngôn ngữ lập trình bậc cao thành chương trình thực hiện được trên máy tính cụ thể.
 - Chúng ta cần phải có chương trình dịch bởi vì chương trình dịch có chức năng chuyển đổi chương trình được viết bằng ngôn ngữ lập trình bậc cao thành chương trình thực hiện được trên máy cụ thể. Nó nhận đầu vào là chương trình viết bằng ngôn ngữ lập trình bậc cao (chương trình nguồn) là dữ liệu vào (*Input*), thực hiện chuyển đổi sang ngôn ngữ máy (chương trình đích) là kết quả ra (*Output*).

Ngoài ra, chương trình dịch trải qua hai giai đoạn: *phân tích và tổng hợp*. Giai đoạn *phân tích* nhằm phân tích chương trình nguồn về *từ vựng* và *cú pháp*. Giai đoạn *tổng hợp* nhằm tạo ra chương trình đích gồm ba bước, đó là:

 - *Sinh mã trung gian* (chuyển chương trình nguồn về chương trình trung gian);
 - *Tối ưu mã* (chỉnh sửa, tối ưu chương trình trung gian);
 - *Sinh mã* (tạo chương trình đích từ chương trình trung gian đã tối ưu).
3. *Biên dịch* và *thông dịch* khác nhau ở những điểm sau:
 - *Trình biên dịch* duyệt, kiểm tra, phát hiện lỗi, xác định chương trình nguồn

có dịch được không. Dịch toàn bộ chương trình nguồn thành một chương trình đích có thể thực hiện trên máy và có thể lưu trữ lại để sử dụng về sau khi cần thiết.

- *Trình thông dịch* lần lượt dịch từng câu lệnh ra ngôn ngữ máy rồi thực hiện ngay câu lệnh vừa dịch được hoặc thông báo lỗi nếu không dịch được.

4. Các điểm khác nhau giữa tên dành riêng và tên chuẩn, đó là: tên dành riêng không được dùng khác với ý nghĩa xác định, tên chuẩn có thể dùng với ý nghĩa khác.

5. Ba tên đúng theo quy tắc của Pascal và có độ dài khác nhau:

tin_hoc

tin_hoc_2007

hanoi2007

Lưu ý: *Tên* trong Pascal được đặt theo quy tắc sau đây:

- Chỉ bao gồm chữ cái, chữ số và dấu gạch dưới;
- Không bắt đầu bằng chữ số;
- Độ dài theo quy định của trình dịch (Turbo Pascal không quá 127 kí tự, Free Pascal không quá 255 kí tự).

Tuy nhiên, *tên* không nên đặt quá dài hay quá ngắn mà nên đặt sao cho gợi nhớ ý nghĩa đối tượng mang tên đó.

6. Những biểu diễn sau đây không phải là hằng trong Pascal:

Biểu diễn	Diễn giải
c) 6,23	Dấu phẩy phải thay bằng dấu chấm (.)
e) A20	Là tên chưa có giá trị

Chú ý:

Biểu diễn	Diễn giải
g) 4+6	Là biểu thức hằng trong Pascal chuẩn cũng được coi là hằng trong Turbo Pascal (TP)
h) 'C	Sai qui định về hằng xâu: thiếu dấu nháy đơn ở cuối
i) 'TRUE'	Là hằng xâu nhưng không phải là hằng logic

II. BÀI TẬP NÂNG CAO

1. Khi lập trình người sử dụng cần quan tâm đến những yếu tố nào?
2. Chương trình dịch có mấy loại? Đó là những loại nào?
3. Trình thông dịch được thực hiện như thế nào? Nó được áp dụng vào những loại ngôn ngữ nào?
4. Trình biên dịch được thực hiện như thế nào? Nó được áp dụng vào những dịch vụ nào?
5. Chương trình dịch có những chức năng nào?
6. Ngôn ngữ lập trình có mấy thành phần cơ bản? Đó là những thành phần nào?
7. Hãy đưa ra 5 tên đúng, 5 tên sai theo quy tắc của Pascal có độ dài khác nhau.

8. Hãy cho biết những biểu diễn nào dưới đây không phải là biểu diễn hằng trong Pascal và chỉ rõ lỗi trong từng trường hợp:
- a) 255.45 b) +50 c) 7;57 d) '55'
- e) 45B f) 1.05E +20 h) 15-5 i) 'FALSE'
9. Hãy đưa ra 3 đoạn chú thích đúng, 3 đoạn chú thích sai trong Pascal

CHƯƠNG II

CHƯƠNG TRÌNH ĐƠN GIẢN

§3. CẤU TRÚC CHƯƠNG TRÌNH

A. TÓM TẮT LÝ THUYẾT

1. Cấu trúc chung

Cấu trúc chung của một chương trình được mô tả như sau:

[< phần khai báo >]
<phần thân>

2. Các thành phần của chương trình

a) Phần khai báo

Có thể có các khai báo cho: tên chương trình, thư viện, hằng, biến và chương trình con.

- **Khai báo tên chương trình:** có thể có hoặc không. Với Pascal, nếu có, phần khai báo tên chương trình được bắt đầu bằng từ khóa `program`, tiếp đến là tên chương trình:

Program <tên chương trình> ;

Trong đó, *tên chương trình* là tên do người lập trình đặt theo qui định về tên.

Ví dụ:

```
program pt_b2;  
hoặc program bai_toan1;
```

- **Khai báo thư viện:** mỗi ngôn ngữ lập trình thường có sẵn một số thư viện cung cấp một số chương trình thông dụng đã được lập sẵn. Để sử dụng các chương trình đó cần khai báo thư viện chứa nó.

Ví dụ, khai báo thư viện:

- Trong Pascal:

`Uses crt;`

- Trong C++:

`#include <stdio.h>`

Thư viện `crt` hoặc `stdio.h` cung cấp các chương trình có sẵn để làm việc với màn hình văn bản và bàn phím. Chẳng hạn, để xóa những gì có trên màn hình: trong Pascal ta dùng lệnh `clrscr`; còn trong C++ dùng lệnh `clrscr` ();

- **Khai báo hằng:** thường được sử dụng cho những giá trị xuất hiện nhiều lần trong chương trình.

Ví dụ, khai báo hằng:

- Trong Pascal:

```
Const MaxN = 1000;
      PI = 3.1416;
```

- Trong C++:

```
Const int MaxN = 1000;
Const float PI = 3.1416;
```

- **Khai báo biến:** Tất cả các biến dùng trong chương trình đều phải đặt tên và phải khai báo cho chương trình dịch biết để lưu trữ và xử lý. Biến chỉ nhận một giá trị tại mỗi thời điểm thực hiện chương trình được gọi là *biến đơn*.

b) Phần thân chương trình

Thân chương trình được tạo bởi dãy lệnh trong phạm vi của cặp dấu hiệu mở đầu và kết thúc.

Ví dụ, thân chương trình trong Pascal:

```
Begin
    [< Dãy lệnh>]
End;
```

3. Ví dụ chương trình đơn giản

- Ví dụ: Chương trình sau thực hiện việc đưa ra màn hình thông báo “Xin chào các bạn!”.

Trong Pascal	Trong C++
<pre>Program vi_du; Begin Writeln('xin chao cac ban!'); End.</pre>	<pre>#include <stdio.h> main () { printf(" Xin chao cac ban!"); }</pre>
- Phần khai báo chỉ có khai báo tên chương trình gồm tên dành riêng <i>program</i> và tên chương trình là <i>vi_du</i>. - Phần thân chương trình chỉ có một câu lệnh <i>writeln</i>, đưa thông báo ra màn hình.	- Phần khai báo chỉ có một câu lệnh <i>include</i> khai báo thư viện <i>stdio.h</i>. - Phần thân chương trình chỉ có một câu lệnh <i>printf</i> đưa thông báo ra màn hình.

- Chương trình Pascal đơn giản có dạng như cột bên phải của bảng dưới đây:

Giải thích	Cấu trúc chương trình Pascal
------------	------------------------------

Phần khai báo	<pre> program <ten chương trình>; uses <ten các thư viện>; const <ten hằng> = <giá trị của hằng>; var <ten biến> : <kiểu dữ liệu>; (* có thể còn những khai báo khác *) begin </pre>
Phần thân chương trình (từ begin đến end)	<pre> [<dãy lệnh>] end. </pre>

§4. MỘT SỐ KIỂU DỮ LIỆU CHUẨN

A. TÓM TẮT LÝ THUYẾT

- *Dữ liệu* là thông tin đã mã hóa trong máy tính. Dữ liệu sau khi tập hợp lại và xử lý sẽ cho ta thông tin.
- Mỗi ngôn ngữ lập trình thường cung cấp một số kiểu dữ liệu chuẩn cho biết phạm vi giá trị có thể lưu trữ, dung lượng bộ nhớ cần thiết để lưu trữ và các phép toán tác động lên dữ liệu.
- Mỗi kiểu dữ liệu được đặc trưng bởi tên kiểu, miền giá trị, kích thước trong bộ nhớ, các phép toán, các hàm và thủ tục sử dụng chúng.

Một số kiểu dữ liệu chuẩn thường dùng cho các biến đơn trong Pascal, đó là:

➤ **Kiểu nguyên:** Các kiểu nguyên được lưu trữ và kết quả tính toán là số nguyên nhưng có hạn chế về miền giá trị. Tập số nguyên là vô hạn và có thứ tự, đếm được nhưng trong máy tính thì kiểu nguyên là hữu hạn, có thứ tự.

Kiểu	Bộ nhớ lưu trữ một giá trị	Phạm vi giá trị
Byte	1 byte	Từ 0 đến 255
Integer	2 byte	Từ -2^{15} đến $2^{15} - 1$
Word	2 byte	Từ 0 đến $2^{16} - 1$
Longint	4 byte	Từ -2^{31} đến $2^{31} - 1$

➤ **Kiểu thực:** Các kiểu thực được lưu trữ và kết quả tính toán chỉ là gần đúng với sai số không đáng kể, nhưng miền giá trị được mở rộng hơn so với kiểu nguyên. Số thực trong máy tính rời rạc và hữu hạn. Phép toán chứa các toán hạng gồm cả kiểu nguyên và kiểu thực sẽ cho kết quả là kiểu thực.

Kiểu	Bộ nhớ lưu trữ một giá trị	Phạm vi giá trị
real	6 byte	0 hoặc có giá trị tuyệt đối nằm trong phạm vi từ 10^{-38} đến 10^{38}
extended	10 byte	0 hoặc có giá trị tuyệt đối nằm trong phạm vi từ 10^{-4932} đến 10^{4932}

➤ **Kiểu kí tự:** Kiểu kí tự có tập hợp giá trị là các kí tự trong bảng mã ASCII, được dùng khi thông tin là các kí tự, xâu (*string*). Kiểu kí tự là kiểu có thứ tự, đếm được. Kí tự đặc biệt dùng để thể hiện sự ngăn cách giữa hai từ viết liên tiếp trong các văn bản, là dấu cách.

Kiểu	Bộ nhớ lưu trữ một giá trị	Phạm vi giá trị
char	1 byte	256 kí tự trong bộ mã ASCII

➤ **Kiểu logic:** Kiểu logic trong Pascal chỉ có hai giá trị là *true* và *false*, được dùng khi kiểm tra một điều kiện hoặc tìm giá trị của một biểu thức logic. Kiểu logic cũng là kiểu thứ tự đếm được.

Kiểu	Bộ nhớ lưu trữ một giá trị	Phạm vi giá trị
boolean	1 byte	<i>true</i> hoặc <i>false</i>

§5. KHAI BÁO BIẾN

A. TÓM TẮT LÝ THUYẾT

- Trong Pascal, mọi biến trong chương trình đều cần khai báo tên và kiểu dữ liệu của nó (có ngôn ngữ gọi đó là định nghĩa biến). Khai báo biến để cấp phát bộ nhớ cho biến và mỗi biến chỉ được khai báo một lần trong chương trình.

- Trong Pascal, khai báo biến bắt đầu bằng từ khóa *var* có dạng sau:

var < danh sách biến >: < kiểu dữ liệu >;

Trong đó:

- danh sách biến* là một hoặc nhiều tên biến, các tên biến được viết cách nhau bởi dấu phẩy;
- kiểu dữ liệu* thường là một trong các kiểu dữ liệu chuẩn hoặc kiểu dữ liệu do người lập trình định nghĩa.

Sau từ khóa *var* có thể khai báo nhiều danh sách biến khác nhau, tức là cấu trúc:

< danh sách biến >: < kiểu dữ liệu >;

có thể xuất hiện nhiều lần.

- Khai báo biến thường đặt sau khai báo hằng như trong bảng sau:

Giải thích	Cấu trúc chương trình Pascal
Phần khai báo	Program <Tên chương trình>; uses <tên các thư viện>; const <tên hằng> = <giá trị của hằng>; var <tên biến>: <kiểu dữ liệu>; (* có thể còn những khai báo khác *)

- Sau khai báo sẽ có một vùng nhớ dành cho biến này với kích đúng bằng kích thước kiểu của nó để lưu trữ giá trị của biến. Địa chỉ đầu của vùng nhớ này được gọi là địa chỉ của biến. Khai báo biến cũng nhằm đưa tên biến vào danh sách các đối tượng cần quản lý của chương trình.

- Kiểu của biến giúp chương trình dịch biết cách tổ chức lưu trữ, truy cập giá trị của biến và áp dụng các thao tác thích hợp trên biến đó.

- Một số chú ý khi khai báo biến:

- Cần đặt tên biến sao cho gợi nhớ đến ý nghĩa của biến đó. Điều này rất có lợi cho việc đọc, hiểu và sửa đổi chương trình khi cần thiết.

- Không nên đặt tên biến quá ngắn hay quá dài, dễ mắc lỗi khi viết nhiều lần tên biến.

- Khai báo biến cần đặc biệt lưu ý đến phạm vi giá trị của nó.

§6. PHÉP TOÁN, BIỂU THỨC, CÂU LỆNH GÁN

A. TÓM TẮT LÝ THUYẾT

1. Các phép toán

Phép toán	Trong toán học	Trong Pascal
Các phép toán số học với số nguyên	+ (cộng), - (trừ), . (nhân), div (chia nguyên), mod (lấy phần dư).	+, -, *, div, mod
Các phép toán số học với số thực	+ (cộng), - (trừ), . (nhân), : (chia)	+, -, *, /
Các phép toán quan hệ	< (nhỏ hơn), ≤ (nhỏ hơn hoặc bằng), ≥ (lớn hơn hoặc bằng), = (bằng), ≠ (khác)	<, <=, >=, =, <>
Các phép toán logic	¬ (phủ định), ∨ (hoặc), ∧ (và)	not, or, and

2. Biểu thức số học

❖ Trong Pascal, **biểu thức số học** là một biến kiểu số hoặc một hằng số hoặc các biến kiểu số và các hằng số liên kết với nhau bởi một số hữu hạn phép toán số học, các dấu ngoặc tròn (và) tạo thành một biểu thức có dạng tương tự như cách viết trong toán học với những quy tắc sau:

- Chỉ dùng cặp ngoặc tròn để xác định trình tự thực hiện phép toán trong trường hợp cần thiết;
- Viết lần lượt từ trái qua phải;
- Không được bỏ qua dấu nhân (*) trong tích.

❖ Các phép toán được thực hiện theo thứ tự:

- Thực hiện các phép toán trong ngoặc trước;
- Trong dãy các phép toán không chứa ngoặc thì thực hiện từ trái sang phải, theo thứ tự các phép toán nhân (*), chia nguyên (*div*), lấy phần dư (*mod*) thực hiện trước và các phép toán cộng (+), trừ (-) thực hiện sau.

Chú ý:

- Nếu biểu thức chứa một hằng hay biến kiểu thực thì ta có biểu thức số học thực, giá trị của biểu thức cũng thuộc kiểu thực.
- Trong một số trường hợp nên dùng biến trung gian để có thể tránh được việc tính một biểu thức nhiều lần.

3. Hàm số học chuẩn

- **Hàm số học chuẩn** là những hàm tính giá trị những hàm toán học thường dùng trong các ngôn ngữ lập trình.
- Mỗi hàm chuẩn có tên chuẩn riêng. Đối số của hàm là một hay nhiều biểu thức số học và được đặt trong cặp ngoặc tròn (và) sau tên hàm.
- Kết quả của hàm có thể là nguyên hoặc thực hay phụ thuộc vào kiểu của đối số.
- Một số hàm chuẩn thường dùng:

Hàm	Biểu diễn Toán học	Biểu diễn trong Pascal	Kiểu đối số	Kiểu kết quả
Bình phương	x^2	sqr(x)	Thực hoặc nguyên	Theo kiểu của đối số
Căn bậc hai	$\sqrt{x}$	Sqrt(x)	Thực hoặc nguyên	Thực
Giá trị tuyệt đối	$ x $	Abs(x)	Thực	Theo kiểu của đối số
Lôgarit tự nhiên	$\ln x$	Ln(x)	Thực	Thực
Lũy thừa của số e	e^x	Exp(x)	Thực	Thực
Sin	$\sin x$	Sin(x)	Thực	Thực
cos	$\cos x$	Cos(x)	Thực	Thực

4. Biểu thức quan hệ

- Hai biểu thức cùng kiểu liên kết với nhau bởi phép toán quan hệ cho ta một biểu thức quan hệ.

Biểu thức quan hệ có dạng:

<biểu thức 1> <phép toán quan hệ> <biểu thức 2>

- Biểu thức quan hệ được thực hiện theo trình tự:
 - Tính giá trị các biểu thức;
 - Thực hiện phép toán quan hệ.

Kết quả của biểu thức quan hệ là giá trị logic: *true*(đúng) hoặc *false*(sai).

5. Biểu thức logic

- Biểu thức logic đơn giản là biến logic hoặc logic.
- Biểu thức logic là các biểu thức logic đơn giản, các biểu thức quan hệ liên kết với nhau bởi phép toán logic. Giá trị biểu thức logic là *true* hoặc *false*. Các biểu thức quan hệ thường đặt trong cặp ngoặc (và).
- Dấu phép toán **not** được viết trước biểu thức cần phủ định.
- Các phép toán **and** và **or** dùng để kết hợp nhiều biểu thức logic hoặc quan hệ, thành một biểu thức thường được dùng để diễn tả các điều kiện phức tạp.

Ta có bảng giá trị phép toán logic:

A	0	1			
Not A	1	0			
A	0	0	1	1	
B	0	1	0	1	
A and B	0	0	0	1	
A or B	0	1	1	1	

6. Câu lệnh gán

- Lệnh gán trong Pascal có dạng:

<tên biến>:= <biểu thức>;

Trong trường hợp đơn giản, *tên biến* là tên của biến đơn.

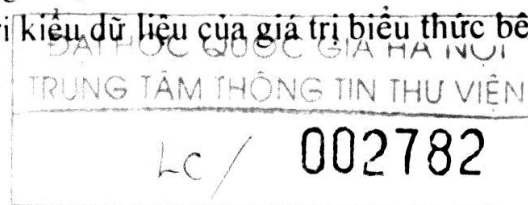
Lệnh gán có chức năng gán giá trị cho một biến, nghĩa là thay giá trị cũ trong ô nhớ (tương ứng với biến) bởi giá trị mới. Giá trị mới là giá trị của một biểu thức. Biểu thức này đã có giá trị xác định thuộc phạm vi của biến. Kiểu giá trị của biểu thức phải phù hợp với kiểu của biến. Một biến chỉ được coi là đã xác định giá trị khi đã nhận được giá trị từ ngoài (đọc từ bàn phím hoặc từ tệp,...) hoặc trực tiếp qua lệnh gán trong chương trình.

Ví dụ:

```
i := i + 1;  
S := S + i;
```

Một số điểm chú ý khi sử dụng lệnh gán:

- Phải viết đúng kí hiệu lệnh gán. Ví dụ, trong Pascal kí tự hai dấu chấm phải viết liền kí tự dấu bằng (:=);
- Biểu thức bên phải cần được xác định giá trị trước khi gán, nghĩa là mọi biến trong biểu thức đã được xác định giá trị và các phép toán trong biểu thức có thể thực hiện được trong miền giá trị của biến.
- Kiểu của biến phải phù hợp với kiểu dữ liệu của giá trị biểu thức bên phải.



§7. CÁC THỦ TỤC CHUẨN VÀO/RA ĐƠN GIẢN

- ❖ Những chương trình đưa dữ liệu vào cho phép đưa dữ liệu từ bàn phím hoặc từ đĩa vào gán cho các biến và những chương trình đưa dữ liệu ra màn hình, giấy và trên đĩa được gọi là *các thủ tục chuẩn vào/ra đơn giản*.
- ❖ Các *thủ tục chuẩn vào/ra đơn giản của Pascal* để nhập dữ liệu vào từ bàn phím và đưa thông tin ra màn hình:

1. Nhập dữ liệu vào từ bàn phím

Việc nhập dữ liệu từ bàn phím được thực hiện bằng thủ tục chuẩn:

`read(<danh sách biến vào>);`

hoặc

`readln(<danh sách biến vào>);`

trong đó *danh sách biến vào* là một hoặc nhiều tên biến đơn. Trong trường hợp nhiều biến thì các tên biến được viết cách nhau bởi *dấu phẩy*.

Ví dụ:

`Read(N);`

`Readln(a, b, c);`

- Khi gặp câu lệnh `read` (hoặc `readln`), chương trình sẽ chờ người dùng nhập giá trị cho danh sách biến và nhấn phím **Enter**, chỉ sau khi nhấn phím **Enter** thì việc nhập giá trị cho danh sách biến mới kết thúc và thực hiện lệnh tiếp theo.
- Khi nhập giá trị cho danh sách biến phải chú ý các giá trị được nhập có kiểu tương ứng với các biến trong sách, giữa hai giá trị liên tiếp phải gõ phím **Space** hoặc phím **Enter**.
- Việc nhập giá trị của biến từ bàn phím được kết thúc bởi việc nhấn phím **Enter** nên không phân biệt `read` và `readln`. Do đó, khi nhập từ bàn phím nên dùng `readln`.

2. Đưa dữ liệu ra màn hình

- Để đưa dữ liệu ra màn hình được thực hiện bằng thủ tục chuẩn:

`write(<danh sách kết quả ra>);`

hoặc

`writeln(<danh sách kết quả ra>);`

trong đó, *danh sách kết quả ra* có thể là tên biến đơn, biểu thức hoặc hằng. Các hằng xâu thường được dùng để tách các kết quả hoặc đưa ra chú thích. Các thành phần trong kết quả ra được viết cách nhau bởi *dấu phẩy*. Với thủ tục `write`, sau khi đưa

các kết quả ra màn hình, con trỏ không chuyển xuống dòng tiếp theo. Với thủ tục *writeln*, sau khi đưa thông tin ra màn hình, con trỏ sẽ xuống đầu dòng tiếp theo.

Ví dụ

Đề nhập giá trị cho biến *M* từ bàn phím, người ta thường cập thu tục:

```
Write('Hay nhap gia tri M: ');  
Readln(M);
```

- Đề chương trình được sử dụng một cách thuận tiện, khi nhập giá trị từ bàn phím cho biến, ta nên có thêm xâu kí tự nhắc nhở giá trị cho biến nào, kiểu dữ liệu gì...

Chú ý:

- ◊ Các thủ tục *readln* và *writeln* có thể không có tham số;
- ◊ Trong thủ tục *write* hoặc *writeln*, sau mỗi kết quả ra (biến, hằng, biểu thức) có thể có *quy cách ra*. Quy cách ra có dạng:
 - Đối với kết quả thực:

:<độ rộng>: <số chữ số thập phân>

- Đối với các kết quả khác:

:<độ rộng>

trong đó: *độ rộng* và *số chữ số thập phân* là các hằng nguyên dương.

§8. SOẠN THẢO, DỊCH, THỰC HIỆN VÀ HIỆU CHỈNH CHƯƠNG TRÌNH

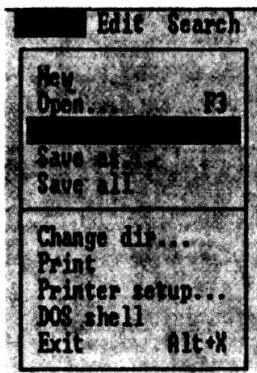
- ❖ Khi làm việc với Turbo Pascal, trong máy tính cần có các tệp, đó là: **turbo.exe, turbo.tpl, graph.tpu, egavga.bgi.**
- ❖ Màn hình làm việc của *TP* có thể như hình dưới đây:



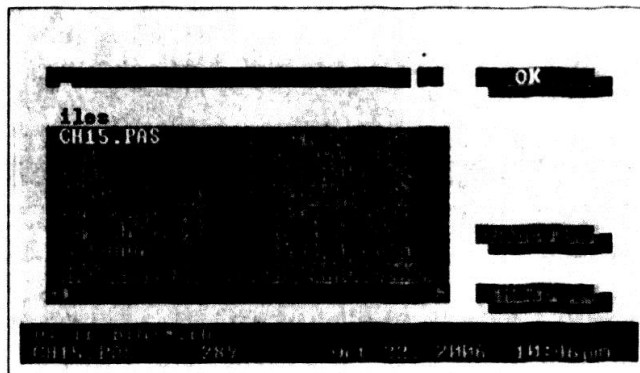
Hình 1. Màn hình làm việc của Pascal

- Nhấp chuột lên mục **File** hoặc nhấn phím **F2** (Hình 2); Hộp thoại *Save File As* xuất hiện. *Xử lý hộp thoại:*

- Gõ tên tệp **PTB2** vào mục *Save file as*
- Gõ phím **Enter** hoặc nhấn phím **OK**.



Hình 2. Chọn File/F2

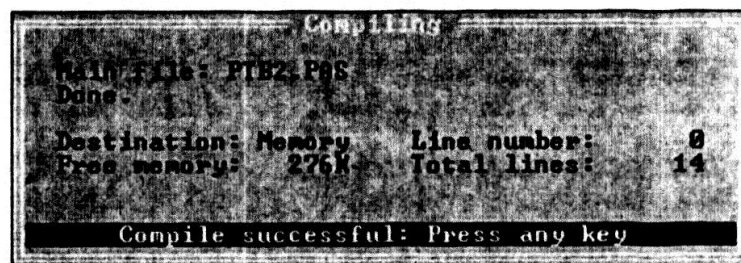


Hình 3. Hộp thoại *Save File As* đặt tên tệp

c) Dịch và sửa lỗi cú pháp (nếu có): nhấn tổ hợp phím **Alt + F9**. Khi dịch chương trình, nếu thấy có thông báo lỗi, các bạn nên xem *phụ lục 7*: Một số thông báo lỗi ở cuối giáo trình.

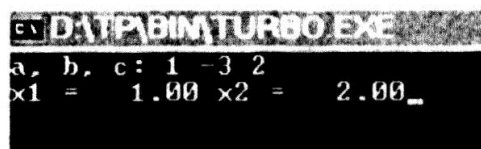
Với chương trình trên, khi dịch sẽ có thông báo lỗi *Error 36: BEGIN expected* (nghĩa là thiếu BEGIN). Lỗi ở dòng thứ 2: `user crt;` được sửa lại thành `uses crt;`

Sau khi dịch xong chương trình, trên màn hình xuất hiện thông báo “*Đã dịch thành công: hãy gõ một phím bất kỳ*” để tiếp tục (Hình 4).



Hình 4. Dịch xong chương trình

d) Thực hiện chương trình: Nhấn tổ hợp phím **Ctrl + F9**. Trên màn hình xuất hiện thông báo nhập vào các giá trị *a, b, c*. Sau khi nhập xong các giá trị $a = 1$, $b = -3$, $c = 2$ thì kết quả của chương trình là $x1 = 1.00$ $x2 = 2.00$ (Hình 5).



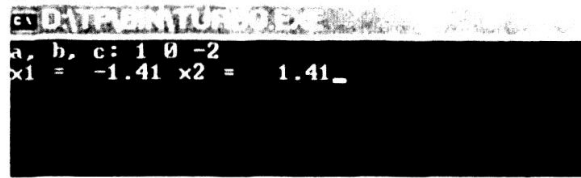
Hình 5. Kết quả của chương trình

Lưu ý:

- Khi nhập các giá trị *a, b, c* ta nên gõ dấu cách sau mỗi lần nhập một giá trị.

- Để quay trở lại màn hình soạn thảo chương trình, ta gõ phím **Enter**. Muốn nhập vào các giá trị khác của **a, b, c** thì ta phải chạy lại chương trình.

e) Nhấn tổ hợp phím **Ctrl + F9** rồi nhập các giá trị 1 0 -2. Kết quả trên màn hình sẽ là $x_1 = -1.41$ $x_2 = 1.41$ (Hình 6).



Hình 6

f) Ta có thể sửa lại chương trình trên khi không dùng biến trung gian **D**:
Chương trình đó là:

```
program Giai_PTB2 ;
uses crt ;
var a, b, c: real ;
    x1, x2: real ;
begin
    clrscr ;
    write('a, b, c: ') ;
    readln(a, b, c) ;
    x1:= (-b - sqrt(b*b - 4*a*c))/(2*a) ;
    x2:= -b/a - x1 ;
    write('x1 = ', x1:6:2, ' x2 = ', x2:6:2) ;
    readln ;
end.
```

➤ Khi nhập các bộ dữ liệu 1; -3; 2 và 1; 0; -2 thì kết quả của chương trình

không có gì thay đổi so với khi dùng biến trung gian **D** ($x_1 = 1.00$ $x_2 = 2.00$ và $x_1 = -1.41$ $x_2 = 1.41$).

g) Sửa lại chương trình để tính nghiệm x_2 bằng hai cách:

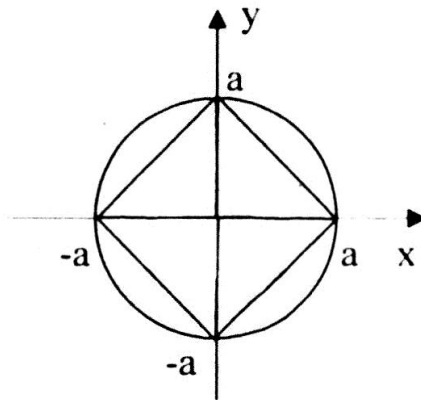
Cách 1: Thay công thức $x_2 := -b/a - x_1$; bằng công thức:

$$x_2 := (-b + \sqrt{b*b - 4*a*c}) / (2*a);$$

Khi đó chương trình sẽ là:

```
Program Giai_PTB2 ;
uses crt;
var
    a, b, c: real ;
    x1, x2: real ;
Begin
    clrscr;
```

9. Hãy viết chương trình nhập số a ($a > 0$) rồi tính và đưa ra diện tích phần được gạch chéo trong hình (kết quả làm tròn đến bốn chữ số thập phân).



Hình 10

10. Lập trình tính và đưa ra màn hình vận tốc v khi chạm đất của một vật rơi từ độ cao h , biết rằng $v = \sqrt{2gh}$, trong đó g là gia tốc rơi tự do và $g = 9,8 \text{ m/s}^2$. Độ cao h (m) được nhập vào từ bàn phím.

Hướng dẫn trả lời câu hỏi và bài tập

1. Sự khác nhau giữa hằng có đặt tên và biến đó là: Xét về mặt lưu trữ giá trị của hằng và biến trong RAM thì: giá trị trong ô nhớ của hằng có đặt tên là không thay đổi, còn giá trị trong ô nhớ của biến thì có thể thay đổi tại từng thời điểm thực hiện chương trình.

2. Khai báo biến nhằm các mục đích sau:

- Xác định kiểu của biến. Trình dịch sẽ biết cách tổ chức ô nhớ chứa giá trị của biến.
- Đưa tên biến vào danh sách các đối tượng được chương trình quản lí.
- Trình dịch biết cách truy cập giá trị của biến và áp dụng thao tác thích hợp cho biến.

3. Trong Pascal, nếu một biến chỉ nhận giá trị nguyên trong phạm vi từ 10 đến 25532 thì biến đó có thể được khai báo bằng các kiểu dữ liệu: *integer*, *real*, *extended*, *longint*.

4. Trong các khai báo trên thì khai báo ở các câu b và d là đúng, tuy nhiên khai báo của câu d là tốt hơn.

5. Để tính diện tích S của hình vuông có các cạnh A với giá trị nguyên nằm trong phạm vi từ 100 đến 200, thì các khai báo b , c , d là đều đúng. Nhưng khai báo c là tốt nhất và tốn ít bộ nhớ cần lưu trữ.

6. Biểu thức Toán học đã cho được viết trong Pascal như sau:

$$(1+z)*(x+y/z)/(a-1/(1+x*x*x))$$

7. Chuyển các biểu thức trong Pascal đã cho sang biểu thức toán học tương ứng:

$$\text{a) } \frac{a^2}{b^2} = \frac{2a}{b}; \quad \text{b) } \frac{(ab)c}{2} = \frac{abc}{2}; \quad \text{c) } \frac{a}{c} = \frac{b}{ac}; \quad \text{d) } \frac{b}{\sqrt{a^2 + b}}$$

8. Biểu thức logic cho kết quả *true* khi tọa độ (x,y) là điểm nằm trong vùng gạch chéo kê cả biên của các hình 2.s và 2.b.

$$((y < -1) \text{ or } (y = 1) \text{ and } ((y = \text{abs}(x)) \text{ or } (y = -\text{abs}(x))))$$

hoặc

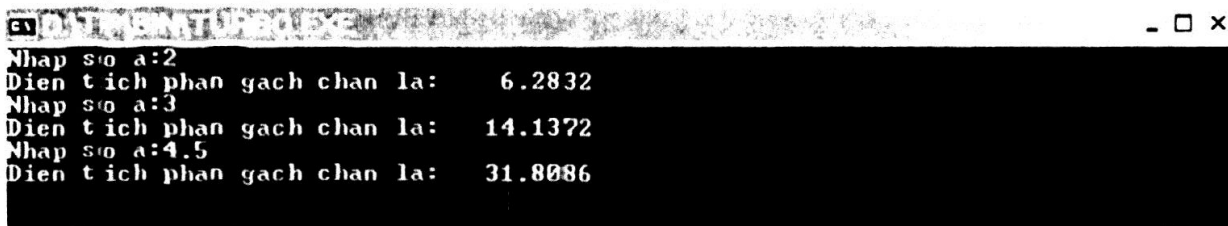
$$(y < -1) \text{ and } (y > = \text{abs}(x))$$

9. Qua hình vẽ, ta nhận thấy rằng diện tích phần gạch bằng $\frac{1}{2}$ diện tích hình tròn tâm $O(0;0)$, bán kính $R=a$. Ta lại biết rằng, diện tích hình tròn được tính theo công thức: $S=R*R*Pi$; $Pi \approx 3,1416$. Khi đó, chương trình tính diện tích phần gạch là như sau:

```
Program dien_tich_phan_gach;
Uses crt;
Var a: real; Const pi = 3,1416;
Begin
Clrscr;
Write('Nhap ban kinh duong tron a (a>0): ');
Readln(a);
write('Dien tich phan gach cheola: ', a*a*pi/2:20:4);
Readln
End.
```

Khi chạy chương trình, nếu $a = 2$ thì diện tích phần gạch là 6.2832;
 nếu $a = 3$ thì diện tích phần gạch là 14.1372;
 nếu $a = 4.5$ thì diện tích phần gạch là 31.8086.

Kết quả chương trình như hình 11 dưới đây:



Hình 11

10. Chương trình tính và đưa ra màn hình vận tốc v:

```
Program tinh_van_toc;
Uses crt;
Const g = 9.8;
Var w,h: real;
Begin
```

```

write('a, b, c: ') ;
readln(a, b, c) ;
x1:= (-b - sqrt(b*b - 4*a*c))/(2*a) ;
x2:= (- b + sqrt(b*b - 4*a*c))/(2*a) ;
write('x1 = ', x1:6:2, ' x2 = ',x2:6:2) ;
readln ;
End.

```

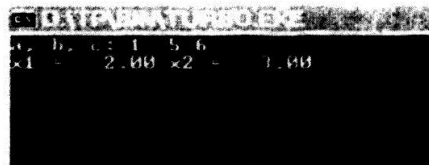
Cách 2: Dùng biến phụ D và thay công thức $x2 := -b/a - x1$; bằng công thức:
 $x2 := (- b + \text{sqrt}(D))/(2*a)$;
 Khi đó chương trình sẽ là:

```

program Giai_PT2 ;
uses crt ;
var a, b, c, D: real ;
    x1, x2: real ;
begin
    clrscr ;
    write('a, b, c: ') ;
    readln(a, b, c) ;
    D:= b*b - 4*a*c ;
    x1:= (-b - sqrt(D))/(2*a) ;
    x2:= (-b + sqrt(D))/(2*a) ;
    write('x1 = ', x1:6:2, ' x2 = ',x2:6:2) ;
    readln ;
end.

```

h) Với bộ dữ liệu 1; -5; 6 thì chương trình đã sửa ở trên ý g) sẽ cho ta kết quả là ($x1 = 2$ $x2 = 3$) (Hình 7)



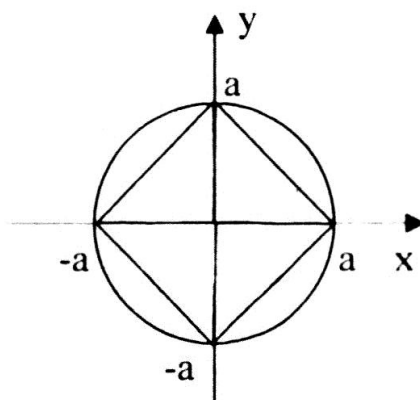
Hình 7

i) Với bộ dữ liệu 1; 1; 1 thì khi dịch chương trình sẽ cho ta kết quả là một thông báo lỗi: Error 207: Invalid floating point operation nghĩa là phép toán với dấu phẩy động không hợp lệ (Hình 8).



Hình 8. Thông báo lỗi 207.

9. Hãy viết chương trình nhập số a ($a > 0$) rồi tính và đưa ra diện tích phần được gạch chéo trong hình (kết quả làm tròn đến bốn chữ số thập phân).



Hình 10

10. Lập trình tính và đưa ra màn hình vận tốc v khi chạm đất của một vật rơi từ độ cao h , biết rằng $v = \sqrt{2gh}$, trong đó g là gia tốc rơi tự do và $g = 9,8 \text{ m/s}^2$. Độ cao h (m) được nhập vào từ bàn phím.

Hướng dẫn trả lời câu hỏi và bài tập

1. Sự khác nhau giữa hằng có đặt tên và biến đó là: Xét về mặt lưu trữ giá trị của hằng và biến trong RAM thì: giá trị trong ô nhớ của hằng có đặt tên là không thay đổi, còn giá trị trong ô nhớ của biến thì có thể thay đổi tại từng thời điểm thực hiện chương trình.

2. Khai báo biến nhằm các mục đích sau:

- Xác định kiểu của biến. Trình dịch sẽ biết cách tổ chức ô nhớ chứa giá trị của biến.
- Đưa tên biến vào danh sách các đối tượng được chương trình quản lý.
- Trình dịch biết cách truy cập giá trị của biến và áp dụng thao tác thích hợp cho biến.

3. Trong Pascal, nếu một biến chỉ nhận giá trị nguyên trong phạm vi từ 10 đến 25532 thì biến đó có thể được khai báo bằng các kiểu dữ liệu: *integer*, *real*, *extended*, *longint*.

4. Trong các khai báo trên thì khai báo ở các câu b và d là đúng, tuy nhiên khai báo của câu d là tốt hơn.

5. Để tính diện tích S của hình vuông có các cạnh A với giá trị nguyên nằm trong phạm vi từ 100 đến 200, thì các khai báo b , c , d là đều đúng. Nhưng khai báo c là tốt nhất và tốn ít bộ nhớ cần lưu trữ.

6. Biểu thức Toán học đã cho được viết trong Pascal như sau:

$$(1+z) * (x+y/z) / (a-1/(1+x*x*x))$$

7. Chuyển các biểu thức trong Pascal đã cho sang biểu thức toán học tương ứng:

$$\text{a) } \frac{a}{b} \div \frac{2}{b} = \frac{2a}{b}; \quad \text{b) } \frac{(ab)c}{2} = \frac{abc}{2}; \quad \text{c) } \frac{a}{c} = \frac{b}{ac}; \quad \text{d) } \frac{b}{\sqrt{a^2 + b}}$$

8. Biểu thức logic cho kết quả *true* khi tọa độ (x,y) là điểm nằm trong vùng gạch chéo kê ca biên của các hình 2.s và 2.b.

$$((y < 1) \text{ or } (y = 1) \text{ and } ((y > \text{abs}(x)) \text{ or } (y = \text{abs}(x))))$$

hoặc

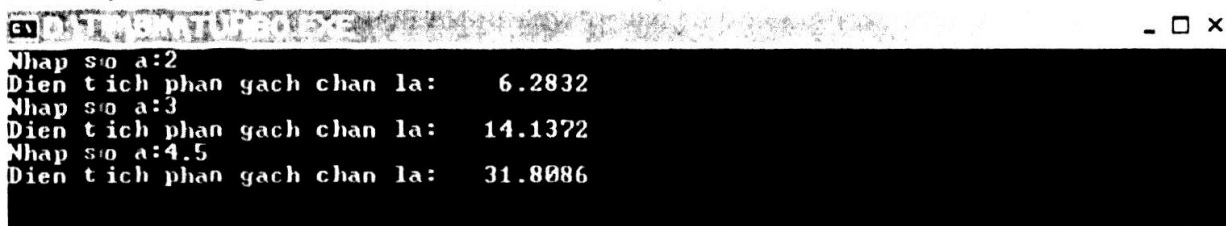
$$(y < -1) \text{ and } (y > = \text{abs}(x))$$

9. Qua hình vẽ, ta nhận thấy rằng diện tích phần gạch bằng $\frac{1}{2}$ diện tích hình tròn tâm $O(0;0)$, bán kính $R=a$. Ta lại biết rằng, diện tích hình tròn được tính theo công thức: $S=R*R*Pi$; $Pi \approx 3,1416$. Khi đó, chương trình tính diện tích phần gạch là như sau:

```
Program dien_tich_phan_gach;
Uses crt;
Var a: real; Conts pi = 3,1416;
Begin
  Clrscr;
  Write('Nhap ban kinh duong tron a (a>0): ');
  Readln(a);
  write('Dien tich phan gach cheola: ', a*a*pi/2:20:4);
  Readln
End.
```

Khi chạy chương trình, nếu $a = 2$ thì diện tích phần gạch là 6.2832;
 nếu $a = 3$ thì diện tích phần gạch là 14.1372;
 nếu $a = 4.5$ thì diện tích phần gạch là 31.8086.

Kết quả chương trình như hình 11 dưới đây:



Hình 11

10. Chương trình tính và đưa ra màn hình vận tốc v:

```
Program tinh_van_toc;
Uses crt;
Const g = 9.8;
Var w,h: real;
Begin
```

```

Clrscr;
Write('Nhap vao do cao h= '); readln(h);
v:=sqrt(2*g*h);
write('Van toc khi cham dat la v = ',v:10:2',m/s');
readln
End.

```

Khi chạy chương trình, nếu $h = 0.45 \text{ m}$ thì vận tốc khi chạm đất $v = 2.97 \text{ m/s}$;
 nếu $h = 1 \text{ m}$ thì vận tốc khi chạm đất $v \approx 4.43 \text{ m/s}$;
 nếu $h = 1.5 \text{ m}$ thì vận tốc khi chạm đất $v \approx 5.42 \text{ m/s}$;
 nếu $h = 2 \text{ m}$ thì vận tốc khi chạm đất $v \approx 6.26 \text{ m/s}$;
 nếu $h = 3 \text{ m}$ thì vận tốc khi chạm đất $v \approx 7.67 \text{ m/s}$;

Kết quả chương trình như hình 12 dưới đây:

```

Nhap do cao h= .45
Van toc luc cham dat la v= 2.9698m/s

Nhap do cao h= 1
Van toc luc cham dat la v= 4.4272m/s

Nhap do cao h= 1.5
Van toc luc cham dat la v= 5.4222m/s

Nhap do cao h= 2
Van toc luc cham dat la v= 6.2610m/s

Nhap do cao h= 3
Van toc luc cham dat la v= 7.6681m/s

```

Hình 12

II. BÀI TẬP NÂNG CAO

- Hãy nêu sự giống nhau và khác nhau của các cặp thủ tục sau:
 - read và readln
 - write và writeln
- Để nhập giá trị cho một biến nào đó từ bàn phím, người ta thường dùng cặp thủ tục nào?
- Hãy nêu một số thao tác và phím tắt thường dùng để soạn thảo và thực hiện một chương trình trên ngôn ngữ lập trình Pascal?
- Hãy viết biểu thức trong toán học dưới đây trong Pascal:

$$(5x^2 + 2x - 3) \sqrt{\frac{2x-1}{x+1}}$$

- Hãy chuyển các biểu thức trong Pascal đã cho sang biểu thức toán học tương ứng:
 - $a*b+c/2$
 - $(a*b)/c/2$
 - $\text{sqrt}((p*(p-a)*(p-b)*(p-c)))$
 - $1/c/a*b$

CHƯƠNG III

CÁU TRÚC Rẽ NHÁNH VÀ LẶP

§9. CẤU TRÚC Rẽ NHÁNH

A. TÓM TẮT LÝ THUYẾT

1. Rẽ nhánh (cấu trúc rẽ nhánh)

- Mệnh đề thiếu: *Nếu ... thì ...*
- Mệnh đề đủ: *Nếu ... thì ..., nếu không thì ...*

2. Câu lệnh *if - then*

a) Dạng thiếu

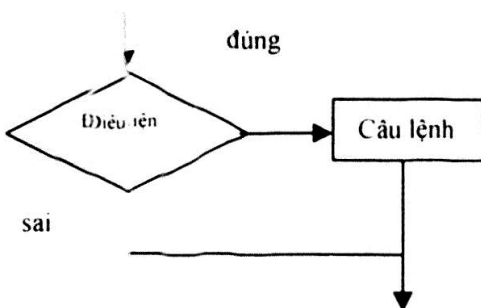
If <điều kiện> then <câu lệnh>;

b) Dạng đủ

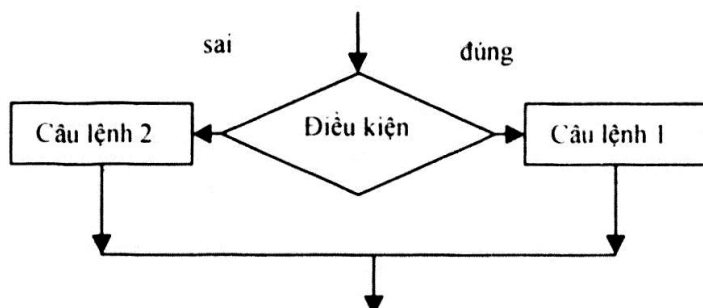
If <điều kiện> then <câu lệnh 1> else <câu lệnh 2>;

Trong đó

- *Điều kiện*: biểu thức quan hệ hoặc logic.
- *Câu lệnh, câu lệnh 1, câu lệnh 2* là một câu lệnh của Pascal.



Hình 13. Dạng thiếu



Hình 14. Dạng đủ

*) Ở **dạng thiếu**: *điều kiện* sẽ được tính và kiểm tra. Nếu *điều kiện* đúng thì *câu lệnh* sẽ được thực hiện, ngược lại thì *câu lệnh* sẽ bị bỏ qua (Hình 13).

*) Ở **dạng đủ**: *điều kiện* cũng được tính và kiểm tra. Nếu *điều kiện* đúng thì *câu lệnh 1* sẽ thực hiện, ngược lại thì *câu lệnh 2* sẽ được thực hiện (Hình 14).

3. Câu lệnh ghép

- *Câu lệnh ghép* là dãy liên tiếp nhiều câu lệnh được ghép lại thành một nhóm nằm giữa hai từ khóa “Begin” và “End” (trong Pascal):

```
Begin
    <các câu lệnh>
End;
```

Ví dụ:

```
Begin
    tg:= a; a:= b; b:= tg;
End;
```

- Việc thực hiện một câu lệnh ghép là thực hiện lần lượt các câu lệnh thành phần trong dãy. Một câu lệnh ghép có thể chứa một câu lệnh ghép khác như một thành phần của nó.

§10. CẤU TRÚC LẶP

A. TÓM TẮT LÝ THUYẾT

1. Lặp

- *Cấu trúc lặp* là một điều khiển thực hiện công việc lặp đi lặp lại khi chưa đủ số lần lặp hoặc khi một điều kiện nào đó còn đúng.
- Quá trình lặp không thể dừng được gọi là *quá trình lặp vô hạn*. Điều này xảy ra khi điều kiện để dừng lặp không còn bị biến đổi giá trị sau mỗi lần lặp. Khi đó để thoát lặp vô hạn, cần có các câu lệnh cho phép thoát ngay khỏi lặp.
- Có hai loại cấu trúc lặp: *lặp với số lần biết trước* và *lặp với số lần chưa biết trước*.

2. Lặp có số lần biết trước và câu lệnh for-do

- Dạng lặp với số lần biết trước dùng để thực hiện *câu lệnh* một số lần xác định. Dạng này dùng một *biến điều khiển* vòng lặp. Trong Pascal mỗi lần thực hiện *câu lệnh* thì *biến điều khiển* (giả sử là *i*) được tự động tăng (nhận giá trị tiếp theo là *succ(i)*) hoặc giảm (nhận giá trị nhỏ hơn ngay trước *pred(i)*). Đến khi biến điều khiển đạt giá trị xác định thì vòng lặp kết thúc.

- Câu lệnh for — do với hai dạng tiến và lùi:

➤ *Dạng lặp tiến:*

for <biến đếm>:= <giá trị đầu> to <giá trị cuối> do <câu lệnh> ;

➤ *Dạng lặp lùi:*

for <biến đếm>:= <giá trị cuối> downto <giá trị đầu> do <câu lệnh> ;

Trong đó:

- *Biến đếm* là biến đơn, thường có kiểu nguyên.
- *Giá trị đầu, giá trị cuối* là các biểu thức cùng kiểu với biến đếm và *giá trị đầu* phải nhỏ hơn hoặc bằng *giá trị cuối*. Nếu *giá trị đầu* lớn hơn *giá trị cuối* thì vòng lặp không được thực hiện.

• **Hoạt động của lệnh `for` – do :**

- Ở dạng **lặp tiến**, *câu lệnh* viết sau từ khóa *do* được thực hiện tuần tự, với *biến đếm* lần lượt nhận giá trị từ *giá trị đầu* đến *giá trị cuối*.

Hoặc hoạt động của dạng lặp tiến có thể được diễn giải như sau:

Bước 1: Biến điều khiển nhận *giá trị đầu*.

Bước 2: Nếu giá trị *biến điều khiển* nhỏ hơn *giá trị cuối* thì chuyển đến *bước 4*.

Bước 3: {giá trị biến điều khiển bằng giá trị cuối} thực hiện câu lệnh, sau đó dừng lặp, chuyển tới câu lệnh tiếp theo vòng lặp.

Bước 4: Thực hiện *câu lệnh* sau *do* và tăng *biến điều khiển* tới giá trị tiếp theo.

- Ở dạng **lặp lùi**, *câu lệnh* viết sau từ khóa *do* được thực hiện tuần tự, với *biến đếm* lần lượt nhận giá trị từ *giá trị cuối* đến *giá trị đầu*.

Ở dạng *lặp lùi* này giá trị của *biến điều khiển* được tự động giảm xuống giá trị tiếp theo sau mỗi lần lặp.

Lưu ý: Trong vòng lặp không được chứa lệnh làm thay đổi giá trị của *biến điều khiển* vì sẽ gây ra tình trạng khó theo dõi và quản lí vòng *for-do*.

3. Lặp với số lần chưa biết trước và câu lệnh `while-do`

- Lặp với số lần chưa biết trước có hai dạng:

Dạng 1: Trong khi *<điều kiện>* còn đúng thì còn thực hiện *<công việc>*;

Dạng 2: Còn thực hiện *<công việc>* trong khi *<điều kiện>* còn đúng.

Trong dạng 1, đầu tiên kiểm tra và tính giá trị của *điều kiện*, nếu *điều kiện* nhận giá trị *true* thì thực hiện *công việc* (một lần). Mỗi lần thực hiện *công việc* có thể sẽ làm thay đổi giá trị của *điều kiện* nên đến một lúc nào đó điều kiện lặp không còn đúng nữa và cấu trúc lặp sẽ được kết thúc. Ngược lại, nếu khi thực hiện *công việc* không làm thay đổi giá trị của *điều kiện* thì cấu trúc lặp kéo dài mãi (gọi là *vòng lặp vô hạn*). Để thoát khỏi vòng lặp vô hạn, trong *công việc* cần có câu lệnh rẽ nhánh thoát khỏi vòng lặp vô hạn khi thỏa mãn điều kiện rẽ nhánh.

- Trong Pascal, lặp với số lần chưa biết trước là dạng *while-do*

Câu lệnh *while-do* chứa một biểu thức điều kiện để điều khiển thực hiện lặp một câu lệnh đơn hoặc kép.

Cú pháp:

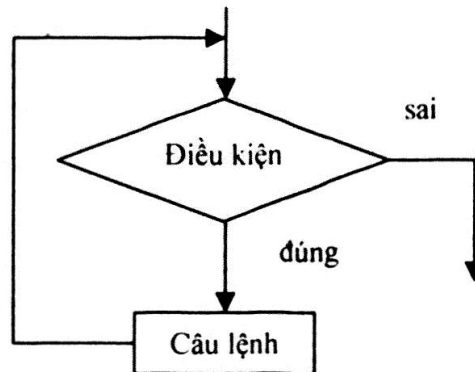
while *<điều kiện>* **do** *<câu lệnh>* ;

Trong đó:

- Điều kiện là biểu thức quan hệ hoặc logic;
- Câu lệnh là một câu lệnh của Pascal.

➤ **Hoạt động của câu lệnh while-do:**

Câu lệnh viết sau từ khóa *do* được thực hiện khi biểu thức *điều kiện* còn nhận giá trị *true*. Biểu thức *điều kiện* được tính giá trị trước khi *câu lệnh* được thực hiện, nhưng nếu biểu thức *điều kiện* đã nhận giá trị *false* ngay từ đầu thì *câu lệnh* không được thực hiện lần nào. Nếu biểu thức *điều kiện* luôn nhận giá trị *true* thì *câu lệnh* được thực hiện mãi, ta gọi là vòng lặp vô hạn.



Hình 15. Sơ đồ lặp với số lần lặp chưa biết trước

❖ **Định lý Bohn Jacopini:** Mọi quá trình tính toán đều có thể mô tả và thực hiện dựa trên ba cấu trúc cơ bản là cấu trúc tuần tự, cấu trúc rẽ nhánh và cấu trúc lặp.



Bài tập và thực hành 2

1. Mục đích, yêu cầu

- Xây dựng chương trình có sử dụng cấu trúc rẽ nhánh;
- Làm quen với việc hiệu chỉnh chương trình.

2. Nội dung

Bài toán: Bộ số *Pi-ta-go*

Biết rằng bộ số nguyên dương a, b, c được gọi là bộ số *Pi-ta-go* nếu tổng các bình phương của hai số bằng bình phương của số còn lại. Viết chương trình nhập từ bàn phím ba số nguyên dương a, b, c và kiểm tra xem chúng có là bộ số *Pi-ta-go* hay không.

Ý tưởng: Kiểm tra xem có đẳng thức nào trong ba đẳng thức sau đây là đúng hay không:

$$a^2 = b^2 + c^2$$

$$b^2 = a^2 + c^2$$

$$c^2 = a^2 + b^2$$

Những công việc cần thực hiện:

a) Gõ chương trình sau:

```
program Pi_ta_go;
uses crt;
var a, b, c: integer;
    a2, b2, c2: longint;
begin
  clrscr;
  write('a, b, c: ');
  readln(a, b, c);
  a2:= a;
  b2:= b;
  c2:= c;
  a2:= a2*a;
  b2:= b2*b;
  c2:= c2*c;
  if (a2 = b2 + c2) or (b2 = a2 + c2) or (c2 = a2 + b2)
  then writeln('Ba so da nhap la bo so Pi - ta - go') else
  writeln('Ba so da nhap khong la bo so Pi-ta-go);
  readln
End.
```

Chú ý:

Trước *else* không có dấu chấm phẩy (;).

b) Lưu chương trình với tên **PITAGO** lên đĩa.

c) Gõ phím **F7** để thực hiện từng câu lệnh chương trình, nhập các giá trị $a = 3$, $b = 4$, $c = 5$.

d) Vào bảng chọn **Debug** mở cửa sổ hiệu chỉnh để xem giá trị $a2$, $b2$, $c2$.

e) Gõ phím **F7** để thực hiện các câu lệnh tính những giá trị nói trên, so sánh với kết quả $a2 = 9$, $b2 = 16$, $c2 = 25$.

f) Quan sát quá trình rẽ nhánh.

g) Lập lại các bước trên với bộ dữ liệu $a = 700$, $b = 1000$, $c = 800$.

h) Nếu thay dãy lệnh

```
a2:= a;
b2:= b;
c2:= c;
a2:= a*a;
b2:= b*b;
c2:= c*c;
```

bằng dãy lệnh

```
a2:= a*a;  
b2:= b*b;  
c2:= c*c;
```

thì kết quả có gì thay đổi với bộ dữ liệu đã cho ở câu g?

Hướng dẫn làm bài tập và thực hành 2

1. Mục đích, yêu cầu

- Xây dựng chương trình có sử dụng cấu trúc rẽ nhánh;
- Làm quen với việc hiệu chỉnh chương trình.

2. Nội dung

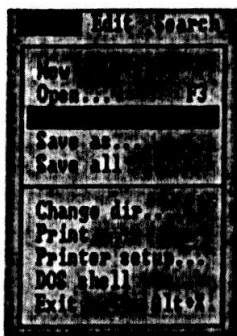
a) Gõ chương trình đã cho ở trên:

- Khởi động Turbo Pascal: Nhấp chuột lên biểu tượng Pascal trên màn hình hoặc lên tệp **Turbo.exe** trong thư mục **BIN** của thư mục **TP** ở ổ đĩa C hoặc ổ đĩa D.
- Gõ nhập chương trình đã cho giống như soạn thảo văn bản trong màn hình làm việc của Pascal.

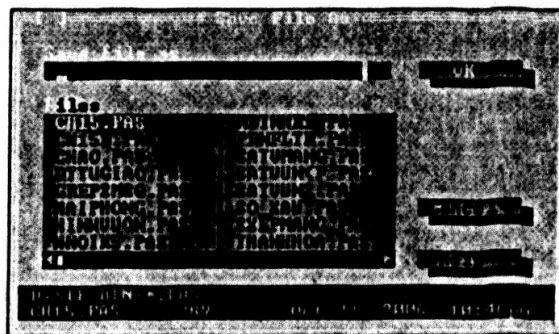
b) Lưu chương trình vào đĩa với tên là **PITAGO.PAS** bằng các thao tác sau:

- Nhấp chuột lên mục **File** hoặc nhấn phím **F2** (Hình 16); Hộp thoại *Save File As* xuất hiện. *Xử lý hộp thoại*:

- Gõ tên tệp **PITAGO** vào mục *Save file as*
- Gõ phím **Enter** hoặc nhấn phím **OK**.



Hình 16. Chọn File/F2

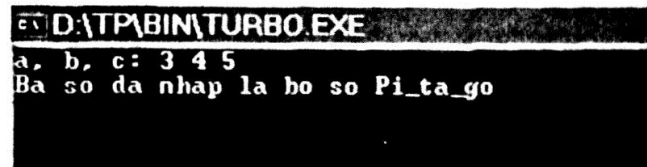


Hình 17. Hộp thoại Save File As đặt tên tệp.



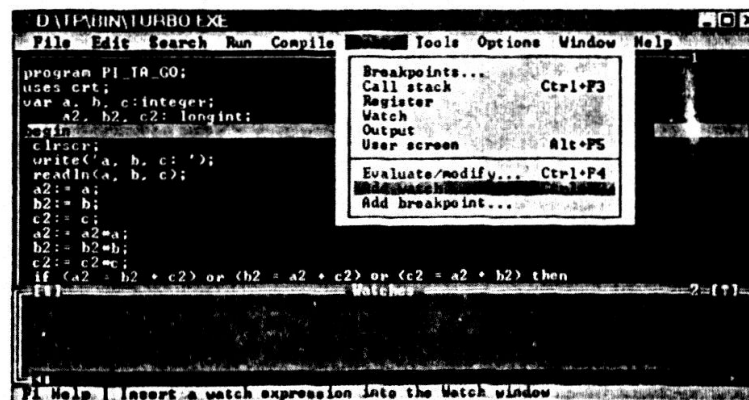
Hình 18. Chương trình

c) Gõ phím **F7** để thực hiện từng câu lệnh chương trình. Đến lệnh `readln(a, b, c)` thì khi nhập các giá trị $a = 3, b = 4, c = 5$ ta nên gõ dấu cách sau mỗi giá trị và kết quả của chương trình sau khi thực hiện trên màn hình sẽ là: “Ba số đã nhập là họ số Pi-ta-go”(Hình 19).



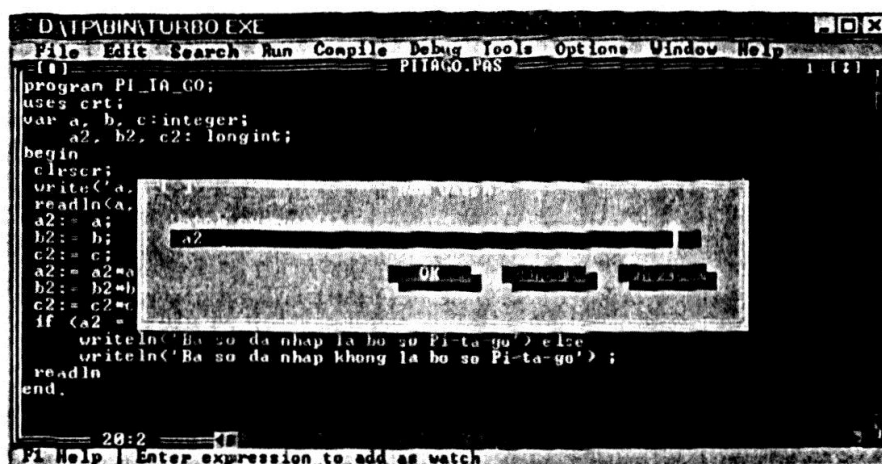
Hình 19. Kết quả của chương trình

d) Sau khi thực hiện xong chương trình, nhập vào các giá trị a, b, c và cho kết quả tốt (câu b). Để vào bảng chọn **Debug** mở cửa sổ hiệu chỉnh để xem giá trị $a2, b2, c2$, ta nhấp chuột vào mục **Debug** (hoặc sử dụng tổ hợp phím **Alt + D**). Tiếp đến để hiện cửa sổ **watches**, ta nhấp chuột chọn mục **Add watch..** hoặc dùng tổ hợp phím **Ctrl + F7** (Hình 20).



Hình 20. Bảng chọn Debug và Add watch...

Khi đó, xuất hiện cửa sổ **Add watch**. Trong cửa sổ **Add watch** ta nhập giá trị $a2$ vào mục **Watch expression** và nhấp chuột chọn **OK** hoặc gõ phím **Enter**.



Hình 21. Cửa sổ Add watch..

Kết quả biến $a2$ sẽ hiện ra trong cửa sổ **watches** và bảng 9. Tương tự, ta nhấp chọn mục **Debug**, sau đó chọn **Add watch...** và nhập vào biến $b2$, nhấp chọn **OK**. Kết quả biến $b2 = 16$ sẽ hiện tiếp dưới biến $a2$. Cuối cùng, biến $c2$ cũng làm tương tự như trên

và kết quả của ba biến $a2 = 9$, $b2 = 16$, $c2 = 25$ được hiện ra trong cửa sổ *watches* (Hình 22).



Hình 22. Giá trị $a2$, $b2$, $c2$ trong cửa sổ *watches*.

Lưu ý: Trong quá trình thực hiện, ta có thể kết hợp sử dụng tổ hợp phím **Ctrl + F5** (thay đổi kích thước cửa sổ hiện thời chứa con trỏ màn hình) và phím **F6** (chuyển cửa sổ hiện thời) để các cửa sổ lộ ra phần thông tin cần theo dõi (không bị cửa sổ khác che khuất).

e) Tiếp tục gõ phím **F7** để theo dõi quá trình rẽ nhánh, ta nhận thấy rằng khi ấn phím **F7** con trỏ ở dòng từ khóa *begin*, tiếp tục ấn cho đến dòng *readln(a, b, c)* thì hiện ra cửa sổ yêu cầu nhập vào ba giá trị của $a = 3$, $b = 4$, $c = 5$. Sau khi nhập vào ba giá trị của ba biến a , b , c ta lại ấn phím **F7**. Cho đến gặp câu lệnh in kết quả chương trình ra màn hình “*Ba số đã nhập là ba số Pi-ta-go*”, rồi kết thúc chương trình.

Vào bảng chọn **Debug** mở cửa sổ hiệu chỉnh để xem giá trị $a2$, $b2$, $c2$ như đã làm ở trên thì ta nhận thấy rằng: với những giá trị của a , b , c như trên thì kết quả của ba biến $a2$, $b2$, $c2$ đều không thay đổi và lần lượt bằng 9, 16 và 25.

f) Quan sát quá trình rẽ nhánh, ta nhận thấy rằng nếu bộ dữ liệu nhập vào thỏa mãn điều kiện “*Tổng các bình phương của hai số bằng bình phương của số còn lại*” thì chương trình sẽ in ra thông báo “*Ba số đã nhập là ba số Pi-ta-go*”, ngược lại chương trình sẽ in ra thông báo “*Ba số đã nhập không là ba số Pi-ta-go*”. Đây chính là dạng *lập đầy đủ*.

g) Khi ta nhập vào bộ dữ liệu mới $a = 700$, $b = 1000$, $c = 800$, thì kết quả của chương trình xuất hiện trên màn hình là: “*Ba số đã nhập không là ba số Pi-ta-go*”.

Vào bảng chọn **Debug** mở cửa sổ hiệu chỉnh để xem giá trị $a2$, $b2$, $c2$, thì ta nhận thấy kết quả của chúng theo thứ tự là 490000, 1000000, 640000 (Hình 23).



Hình 23

h) Nếu thay dãy lệnh

```
a2 := a;  
b2 := b;  
c2 := c;  
a2 := a*a;  
b2 := b*b;  
c2 := c*c;
```

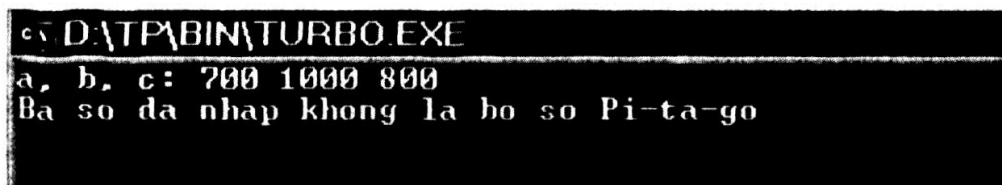
bằng dãy lệnh

```
a2 := a*a;  
b2 := b*b;  
c2 := c*c;
```

thì chương trình mới sẽ là như sau:

```
program Pi_ta_go;  
uses crt;  
var a, b, c: integer;  
    a2, b2, c2: longint;  
begin  
  clrscr;  
  write('a, b, c: ');  
  readln(a, b, c);  
  a2 := a*a;  
  b2 := b*b;  
  c2 := c*c;  
  if (a2 = b2 + c2) or (b2 = a2 + c2) or (c2 = a2 + b2)  
  then writeln('Ba so da nhap la bo so Pi - ta - go') else  
  writeln('Ba so da nhap khong la bo so Pi-ta-go');  
  readln  
End.
```

Khi chạy chương trình và nhập vào bộ dữ liệu $a = 700$, $b = 1000$, $c = 800$, kết quả của chương trình sẽ là: “Ba so da nhap khong la bo so Pi-ta-go”(Hình 24).



Hình 24

B. CÂU HỎI VÀ BÀI TẬP

I. BÀI TẬP CƠ BẢN

1. Hãy cho biết sự giống và khác nhau của hai dạng câu lệnh `if-then`.
2. Câu lệnh ghép là gì? Tại sao phải có câu lệnh ghép?

3. Có thể dùng câu lệnh `while`-do để thay cho câu lệnh `for`-do được không? Nếu được, hãy thực hiện điều đó với chương trình.
4. Viết câu lệnh `if` tính:

$$a) \quad z = \begin{cases} x^2 + y^2 & \text{nếu } x^2 + y^2 \leq 1 \\ x + y & \text{nếu } x^2 + y^2 > 1 \text{ và } y \geq x. \\ 0.5 & \text{nếu } x^2 + y^2 > 1 \text{ và } y \leq x. \end{cases}$$

$$b) \quad z = \begin{cases} |x| + |y| & \text{nếu điểm } (x; y) \text{ thuộc hình tròn bán kính } r (r > 0), \text{ tâm } (a; b). \\ x + y & \text{trong trường hợp còn lại.} \end{cases}$$

5. Lập trình tính:

$$a) \quad Y = \sum_{n=1}^{50} \frac{n}{n+1};$$

$$b) \quad e(n) = 1 + \frac{1}{1!} + \frac{1}{2!} + \dots + \frac{1}{n!} + \dots \text{ với } n \text{ lần lượt bằng } 3, 4, \dots \text{ cho đến khi } \frac{1}{n!} < 2 \times 10^{-6}.$$

Đưa các giá trị $e(n)$ ra màn hình.

6. Lập trình để giải bài toán cổ sau:

Vừa gà vừa chó.
Bó lại cho tròn.
Ba mươi sáu con,
Một trăm chân chẵn.
Hỏi bao nhiêu con mỗi loại?

7. Nhập từ bàn phím tuổi của cha và con (tuổi của cha hơn tuổi con ít nhất là 25). Đưa ra màn hình bao nhiêu năm nữa thì tuổi cha gấp đôi tuổi con.
8. Một người gửi tiết kiệm không kì hạn với số tiền 4 đồng với lãi suất 0,2% mỗi tháng. Hỏi sau 1 tháng, người đó rút tiền thì sẽ nhận được số tiền là bao nhiêu. Biết rằng tiền gửi tiết kiệm không kì hạn không được tính lãi kép.

Hướng dẫn trả lời câu hỏi và bài tập

1. Sự giống và khác nhau của hai dạng câu lệnh `if-then`

Hai dạng câu lệnh `if-then` như sau:

- a) Dạng thiếu

If <điều kiện> then <câu lệnh>;

b) Dạng đủ

If <điều kiện> then <câu lệnh 1> else <câu lệnh 2> ;

trong đó:

- *Điều kiện*: biểu thức quan hệ hoặc logic.
- *Câu lệnh, câu lệnh 1, câu lệnh 2* là một câu lệnh của Pascal.
- **Giống nhau**: đều cùng là câu lệnh rẽ nhánh và khi gặp một điều kiện nào đó thì chọn lựa thực hiện thao tác thích hợp.
- **Khác nhau**: Trong câu lệnh *if-then* **dạng thiếu**, nếu điều kiện không đúng thì thoát khỏi cấu trúc rẽ nhánh, thực hiện câu lệnh tiếp theo của chương trình, còn trong câu *if-then* **dạng đủ**, nếu điều kiện không đúng thì thực hiện *công việc 2*, sau đó mới thoát khỏi cấu trúc rẽ nhánh, thực hiện câu lệnh tiếp theo của chương trình.

2. Câu lệnh ghép là một câu lệnh được hợp thành từ nhiều câu lệnh thành phần (đơn hoặc kép). Câu lệnh ghép nhằm thực hiện thao tác gồm nhiều thao tác thành phần. Mỗi thao tác thành phần tương ứng với một câu lệnh đơn hoặc câu lệnh ghép khác. Về mặt ngôn ngữ lập trình, câu lệnh ghép là một trong các yếu tố để tạo khả năng chương trình có cấu trúc.

Câu lệnh ghép trong Pascal:

```
Begin
    <các câu lệnh>
End;
```

3. Có thể thay thế đoạn chương trình chứa câu lệnh *for-do* (dạng lặp tiến)

for <biến đếm>:= <giá trị đầu> to <giá trị cuối> do <câu lệnh> ;

bằng đoạn chương trình chứa câu lệnh *while-do* như sau:

```
i:= <giá trị đầu>;
while <i<= giá trị cuối> do
begin
    <câu lệnh>;
    <tăng i một đơn vị>;
end;
```

Như vậy, chương trình tính **Tong_1a** viết bằng lệnh *for-do*

```
program Tong_1a;
uses crt;
var S:real;
    a, N: integer;
begin
    clrscr;
    write('Hay nhap gia tri a vao!');
    readln(a);
    S:=1.0/a;
```

```

    for N:=1 to 100 do
    S:= S+1.0/(a+N);
    writeln('Tong S la:', S:8:4);
    readln;
End.

```

được viết lại bằng lệnh while-do như sau:

```

Program Tong_1a;
uses crt;
var S:real;
    a, N: integer;
Begin
clrscr;
write('Hay nhap gia tri a vao!');
readln(a);
S:= 1.0/a;
N:= 1;
while N<=100 do
begin
    S:= S + 1.0/(a+N);
    N:= N + 1;
end;
writeln('Tong S la:', S:8:4);
readln;
End.

```

4. Các biểu thức đã cho được viết theo lệnh if như sau:

a)

```

if sqr(x) + sqr(y) <= 1 then z:= sqr(x) + sqr(y)
else
    if y >= x then z:= x + y
    else z:= 0.5;

```

b)

```

if sqr(x-a)+sqr(y-b) <= sqr(r) then z:= abs(x)+abs(y)
else z:= x + y;

```

5. Lập trình tính các biểu thức:

a) $Y = \sum_{n=1}^{50} \frac{n}{n+1}$;

```

program Tong_5a;
uses crt;
var y: real;
    n: byte;

```

```

Begin
clrscr;
y:= 0;
for n:= 1 to 50 do
    y:= y + n/(n+1);
    writeln('Tong y la: ', y:0:18);
readln;
End.

```

Nếu biến y khai báo theo kiểu extended thì chương trình tính tổng y sẽ là như sau:

```

{$e+, N+}
program Tong_5a;
uses crt;
var y: real;
    n: byte;
Begin
clrscr;
y:= 0;
for n:= 1 to 50 do
    y:= y + n/(n+1);
    writeln('Tong y la: ', y:0:18);
readln
End.

```

- b) $e(n) = 1 + \frac{1}{1!} + \frac{1}{2!} + \dots + \frac{1}{n!} + \dots$ với n lần lượt bằng 3, 4, ... cho đến khi $\frac{1}{n!} < 2 \times 10^{-6}$. Đưa các giá trị $e(n)$ ra màn hình.

```

program Tong_5b;
uses crt;
var n: longint;
    e, sh: real;
Begin
clrscr;
sh:= 1/2;
n:= 2;
e:= 2 + sh;
while sh>= 2*1E-16 do
    begin
        inc(n);
        sh:= sh*(1/n);
        e:= e + sh;
    end;
    writeln('Gia tri e(n) la: ', e:10:6);
readln
End.

```

6. Chương trình bài toán cổ như sau:

```
program Tim_ga_cho;
uses crt;
var ga, cho: integer;
Begin
clrscr;
  for cho:= 1 to 24 do
    begin
      ga:= 36 - cho;
      if ga + 2*cho = 50 then
        writeln('Ga:',ga,' Cho:',cho);
    end;
  readln
End.
```

7. Chương trình về tuổi cha và tuổi con:

```
program tuoi_cha_con;
uses crt;
var tuoicha, tuoicon, nam: longint;
begin
clrscr;
write('Nhap tuoi cha va con(tuoicha-tuoicon>=25) :');
readln(tuoicha,tuoicon);
nam:= 0;
while tuoicha<>2*tuoicon do
  begin
    tuoicha:= tuoicha + 1;
    tuoicon := tuoicon + 1;
    nam:= nam +1;
  end;
  writeln('Sau',nam,'nam,tuoi cha gap doi tuoi con');
  readln
End.
```

8. Chương trình gửi tiền tiết kiệm

```
program Gui_tiet_kiem;
uses crt;
const laisuat = 0.002;
var tiengui, tienrutve, luu: real;
    thang : integer;
Begin
clrscr;
  write('Nhap vao so tien gui:');
  readln(tiengui);
```

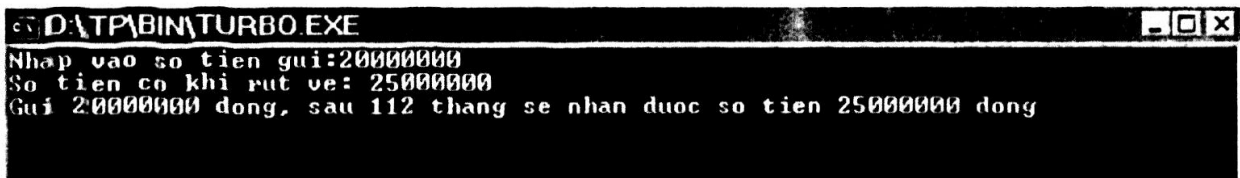
```

luu:= tiengui;
write('So tien co khi rut ve: ');
readln(tienrutve);
thang:= 0;
while tiengui < tienrutve do
    begin
        tiengui:= tiengui + tiengui*laisuat ;
        thang:= thang + 1;
    end;
write('Gui',luu:0:0,' dong, sau ', thang,' thang');
writeln('se nhan duoc so tien',tienrutve:0:0,' dong');
readln
End.

```

Khi chạy chương trình, với số tiền gửi ban đầu là 20000000 đồng. Muốn có được số tiền sau khi rút về là 25000000 đồng thì phải mất 112 tháng.

Kết quả chương trình cho như hình 25 dưới đây:



Hình 25

II. BÀI TẬP NÂNG CAO

1. Lập trình tính:

a) $S_1 = 1 + 1/2 + 1/3 + 1/4 + 1/5 + \dots$

Yêu cầu là quá trình tính sẽ dừng lại khi $2-s < 0,001$.

c) $S_2 = 1 - 1/2 + 1/4 - 1/6 + \dots - 1/102$

2. Cho máy nhận vào ba số nguyên bất kỳ. Ba số đó có thể làm thành cấp số cộng không? (có thể sắp lại thứ tự của chúng).

3. Viết chương trình cho máy nhận vào ba số thực bất kỳ. Xét ba số đó có làm thành ba cạnh của một tam giác không? Nếu có thì tam giác đó là tam giác ba góc đều nhọn, tam giác vuông hay tam giác có một góc tù?

4. Trong đợt quyên góp sắt vụn để gây quỹ các bạn nghèo có hoàn cảnh khó khăn, tổ 3 có 12 bạn, mỗi bạn nộp một khối lượng sắt vụn nào đó bằng kg. Viết chương trình cho máy nhận vào khối lượng sắt vụn của từng bạn và tính tổng khối lượng sắt vụn của cả 12 bạn trong tổ.

5. Viết chương trình giải bài toán 100 con trâu 100 bó cỏ, một con trâu đực ăn 5 bó cỏ, một con trâu nài ăn 3 bó cỏ, 3 con trâu cái ăn 1 bó cỏ. Hỏi mỗi loại trâu có mấy con?

CHƯƠNG IV

Kiểu dữ liệu có cấu trúc

§11. KIỂU MẢNG VÀ BIẾN CÓ CHỈ SỐ

A. TÓM TẮT LÝ THUYẾT

1. Kiểu mảng một chiều

- *Mảng một chiều* là dãy hữu hạn các phần tử cùng kiểu.
- Mảng được đặt tên và mỗi phần tử của nó có một chỉ số.
- Để mô tả mảng một chiều cần xác định kiểu của các phần tử và cách đánh số các phần tử của nó (mỗi phần tử của nó có một chỉ số).
- Để người lập trình có thể xây dựng và sử dụng kiểu mảng một chiều, các ngôn ngữ lập trình có quy tắc cách thức cho phép xác định:
 - Tên kiểu mảng một chiều;
 - Số lượng phần tử;
 - Kiểu dữ liệu của phần tử;
 - Cách khai báo biến;
 - Cách tham chiếu đến phần tử.
- Có thể truy cập (hay thao tác) trên mỗi phần tử của mảng, trong việc làm đó mỗi phần tử của mảng được xác định bởi tên của mảng và chỉ số tương ứng của phần tử này.
- *Kiểu mảng* là một kiểu dữ liệu có cấu trúc, rất cần thiết và hữu ích trong nhiều chương trình.

❖ Khai báo

Khai báo mảng một chiều có dạng tổng quát như sau:

Cách 1: Khai báo trực tiếp biến mảng một chiều:

```
var <tên biến mảng>:= array[kiểu chỉ số] of <kiểu phần tử>;
```

Cách 2: Khai báo gián tiếp biến mảng qua kiểu mảng một chiều:

```
type <tên kiểu mảng>= array[kiểu chỉ số] of <kiểu phần tử>;  
var <tên biến mảng>:<tên kiểu mảng>;
```

Trong đó:

- ❖ *Kiểu chỉ số* thường là một đoạn số nguyên liên tục có dạng $n1..n2$ với $n1, n2$ là các hằng hoặc biểu thức nguyên xác định chỉ số đầu và chỉ số cuối ($n1 < n2$);

❖ *Kiểu phần tử* là kiểu của phần tử mảng.

Ví dụ, các khai báo sau đây là hợp lệ về mảng một chiều:

Type

```
arrayReal = array[-100..200] of real;  
ArrayBoolean = array[-n+1.. n+1] of boolean;  
ArrayInt = array[-100..0] of integer;
```

trong đó, n là hằng nguyên.

Tham chiếu tới phần tử của mảng một chiều được xác định bởi tên mảng cùng với chỉ số, được viết trong cặp dấu ngoặc [và].

❖ **Lưu ý:**

- Đối với ví dụ 1 về bài toán: “Tìm phần tử lớn nhất của dãy số nguyên”.

Chương trình của bài toán như sau:

```
Program tìmMax;  
uses crt;  
const  
    Nmax = 250;  
type  
    arrInt = array[1..Nmax] of integer;  
var  
    N, i, Max, csmax: integer;  
    A: arrInt;  
Begin  
clrscr;  
    write('Nhap so luong phan tu cua day so, N= ');  
    readln(N);  
    for i:= 1 to N do  
        begin  
            write('Phan tu thu ', i, ' = ');  
            readln(A[i]);  
        end;  
    Max:= A[1]; csmax:= 1;  
    for i:= 2 to N do  
        if A[i] > Max then  
            begin  
                Max:= A[i];  
                csmax:= i;  
            end;  
    writeln('Gia tri cua phan tu Max: ', Max);  
    writeln('Chi so cua phan tu Max: ', csmax);  
    readln  
End
```

Kh chạy chương trình, giả sử ta nhập giá trị các phần tử:

Phần tử thứ nhất là 15;

Phần tử thứ hai là 5;

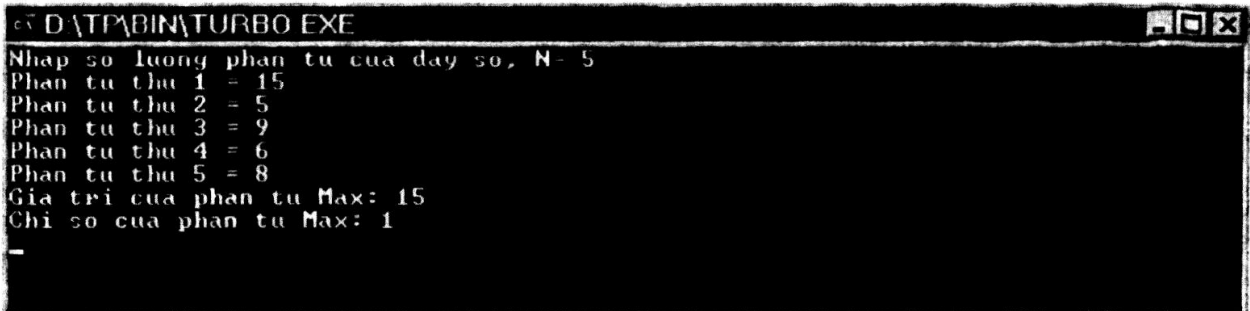
Phần tử thứ ba là 9;

Phần tử thứ tư là 6;

Phần tử thứ năm là 8;

thì trên màn hình thông báo kết quả: "Giá trị của phần tử Max: 15",
"Chỉ số của phần tử lớn nhất là 1".

Khi đó màn hình có dạng như hình 26 dưới đây:



```

D:\TP\BIN\TURBO EXE
Nhập số lượng phần tử của dãy số, N= 5
Phần tử thứ 1 = 15
Phần tử thứ 2 = 5
Phần tử thứ 3 = 9
Phần tử thứ 4 = 6
Phần tử thứ 5 = 8
Giá trị của phần tử Max: 15
Chỉ số của phần tử Max: 1

```

Hình 26. Kết quả chương trình tìm phần tử lớn nhất của dãy số nguyên

Từ chương trình này, chúng ta rút ra được một số điều cơ bản cần phải quan tâm, đó là:

- ◊ Dùng mảng có kiểu phần tử là nguyên để biểu diễn một dãy hữu hạn số nguyên và cách khai báo mảng này.
- ◊ Câu lệnh *for-do* thứ nhất trong chương trình thể hiện một nhiệm vụ trong bước 1 của thuật toán, dùng để nhập các phần tử của mảng. Số phần tử thực sự của mảng do người chạy chương trình nhập vào bởi câu lệnh ngay trước câu lệnh *for-do* này.
- ◊ Câu lệnh *for-do* thứ hai trong chương trình thể hiện vòng lặp (gồm bước 3 và 4 trong thuật toán), dùng để duyệt tuần tự từng phần tử trong mảng lọc lấy phần tử tạm thời là lớn nhất (trong các phần tử đã duyệt qua).
- Đối với ví dụ 2, về bài toán: "Sắp xếp dãy số nguyên bằng thuật toán đổi chỗ". Chương trình của bài toán như sau:

```

program sapxep;
uses crt;
const Nmax = 250;
type
  ArrInt = array[1..Nmax] of integer;
var
  N, i, j, t: integer;
  A: ArrInt;
begin
  clrscr;

```

```

write('Nhap so luong phan tu cua day so, N=');
readln(N);
for i:= 1 to N do
begin
    write('Phan tu thu',i,'=');
    readln(A[i]);
end;
for j:= N downto 1 do
begin
    for i:=1 to j-1 do
    if A[i] > A[i+1] then
    begin
        t:= A[i];
        A[i]:= A[i+1];
        A[i+1]:= t;
    end;
end;
writeln('Day so duoc sap xep la:');
for i:= 1 to N do write(A[i]:4);
readln
End.

```

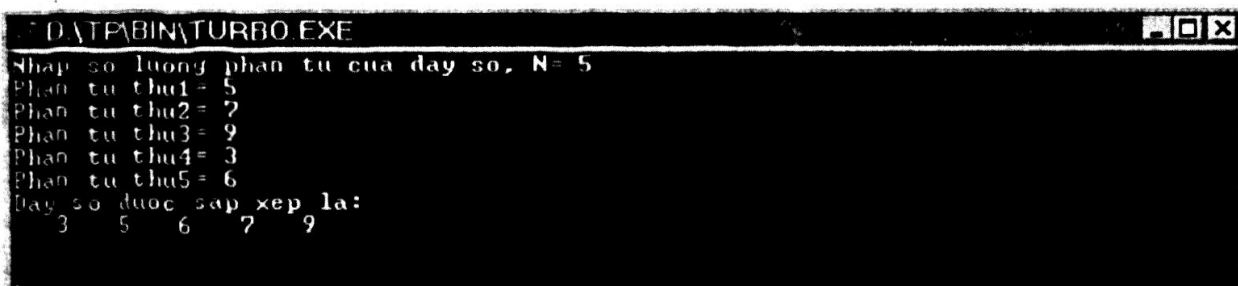
Khi chạy chương trình, giả sử ta nhập số phần tử của dãy là 5 với giá trị các phần tử như sau:

Phần tử thứ nhất= 5;
 Phần tử thứ hai= 7;
 Phần tử thứ ba= 9;
 Phần tử thứ tư= 3;
 Phần tử thứ năm= 6;

thì trên màn hình sẽ có thông báo:

Day so duoc sap xep la:
 3 5 6 7 9

Màn hình kết quả có dạng như hình 27 dưới đây:



Hình 27. Kết quả chương trình sắp xếp dãy số

Từ chương trình này, chúng ta rút ra được một số điều cơ bản cần phải quan tâm, đó là:

- ◊ Khái niệm *lượt*: sau lượt(lần) duyệt thứ nhất giá trị lớn nhất xếp đúng vị trí là ở cuối dãy. Tương tự, sau lượt thứ hai, giá trị lớn thứ hai được xếp ở vị trí sát cuối,... Sau mỗi lượt có ít nhất một số hạng đã xếp đúng vị trí. Trong thuật toán phải thực hiện bao nhiêu lượt như vậy, mỗi lượt thực hiện trên đoạn nào (từ đâu đến đâu) của dãy số? Giá trị của j chính là chỉ số phần tử cuối trong đoạn được duyệt của lượt. Đây chính là câu lệnh `for j := N downto 2` do với biến đếm j chạy từ N về 2.
- ◊ Mỗi lượt bao gồm việc thực hiện một số thao tác: so sánh một phần tử với phần tử đứng ngay sau nó để xử lí, bắt đầu từ phần tử đầu tiên trong dãy đến phần tử thứ j . Thao tác so sánh để quyết định xử lí (tráo đổi hai phần tử) được lặp một số lần. Đây chính là câu lệnh lặp mà chương trình dùng để thể hiện mỗi lượt:

```
for i:=1 to j-1 do
  if A[i] > A[i+1] then
    begin
      t:= A[i];
      A[i]:= A[i+1];
      A[i+1]:= t;
    end;
```

Giá trị biến đếm i chính là chỉ số phần tử được lấy so sánh với phần tử kế sau nó trong dãy số. Trong cấu trúc lặp này, i chỉ lấy đến $j-1$ chứ không phải là đến j .

- Đối với ví dụ 3, về bài toán: “Tìm kiếm nhị phân”.

➤ **Thuật toán:**

Bước 1: Nhập N , các số hạng $a_1, a_2, \dots, a_N$ và khóa k ;

Bước 2: $Dau \leftarrow 1, Cuoi \leftarrow N$;

Bước 3: $Giua \leftarrow \left\lfloor \frac{Dau + Cuoi}{2} \right\rfloor$;

Bước 4: Nếu $A_{giua} = k$ thì thông báo chỉ số $Giua$, rồi kết thúc;

Bước 5: Nếu $A_{giua} > k$ thì đặt $Cuoi = Giua - 1$ rồi chuyển đến bước 7;

Bước 6: Nếu $A_{giua} \leq k$ $Dau \leftarrow Giua + 1$;

Bước 7: Nếu $Dau > Cuoi$ thì thông báo dãy A không có số hạng có giá trị bằng k , rồi kết thúc;

Bước 8: Quay lại bước 3.

Từ thuật toán của bài toán, chúng ta rút ra được một số điều cơ bản cần phải lưu ý, đó là:

- ◊ Mảng đã được sắp xếp tăng dần

- ◊ Trong thuật toán, việc tìm kiếm thực chất là lặp một số lần (chưa xác định được trước) các thao tác sau: chọn số hạng ở “*giữa*” dãy, so sánh số hạng đó với k , căn cứ vào kết quả so sánh này để hoặc kết luận đã tìm thấy (trường hợp xảy ra *bằng*) hoặc thu hẹp phạm vi tìm kiếm (trường hợp *không bằng*).
- ◊ Khi nào quá trình lặp nói trên dừng lại? Quá trình lặp đó cần dừng lại với một trong hai sự kiện sau xảy ra gồm đã tìm thấy hoặc không gian tìm kiếm đã trở nên bằng rỗng (nghĩa là không còn đoạn nào của dãy cho ta hy vọng chứa phần tử cần tìm).
- ◊ Phạm vi tìm kiếm trên dãy là một đoạn được xác định bởi các biến nguyên *Dau* và *Cuoi*, tương ứng cho biết bắt đầu từ phần tử có chỉ số *Dau* của dãy cho đến phần tử có chỉ số *Cuoi* của dãy. Từ đó, ta đưa ra được công thức xác định phần tử ở “*giữa*” phạm vi tìm kiếm và công thức xác định lại giá trị cho biến *Dau* hay *Cuoi* trong mỗi trường hợp thu hẹp phạm vi tìm kiếm.

- Chương trình của bài toán như sau:

```

program TK_nhiphan;
uses crt;
const
  Nmax = 250;
type
  ArrInt = array[1..Nmax] of integer;
var
  N, i, k: integer;
  Dau, Cuoi, Giua: integer;
  A: ArrInt;
  Tim_thay: boolean;
Begin
  clrscr;
  write('Nhap so luong phan tu cua day so, N=');
  readln(N);
  writeln('Nhap cac phan tu cua day so tang:');
  for i:= 1 to N do
    begin
      write('Phan tu thu ', i, '=');
      readln(A[i]);
    end;
  write('Nhap gia tri k =');
  readln(k);
  Dau:= 1; Cuoi:= N; Tim_thay:=false;
  while (Dau <= Cuoi) and not (Tim_thay) do
    begin
      Giua:= (Dau+Cuoi) div 2;
      if A[Giua] = k then
        Tim_thay:= true
    end;
End;

```

```

else
    if A[Giua] > k then Cuoi:= Giua-1
    else Dau:= Giua + 1;
End;
if Tim_thay then writeln('Chi so tim duoc la:', Giua)
else writeln('Khong tim thay');
readln
End.

```

Khi chạy chương trình, giả sử ta nhập số phần tử của dãy $N=5$ với giá trị các phần tử như sau:

Phần tử thứ nhất= 3;
 Phần tử thứ hai= 5;
 Phần tử thứ ba= 7;
 Phần tử thứ tư= 9;
 Phần tử thứ năm= 12;

tiếp đến, ta nhập giá trị $k = 9$ thì trên màn hình sẽ có thông báo:

Chi so tim duoc la: 4

Kết quả trên màn hình có dạng như hình 28 dưới đây:

```

D:\TP\BIN\TURBO.EXE
Nhap so luong phan tu cua day so, N= 5
Nhap cac phan tu cua day so tang:
Phan tu thu 1= 3
Phan tu thu 2= 5
Phan tu thu 3= 7
Phan tu thu 4= 9
Phan tu thu 5= 12
Nhap gia tri k=9
Chi so tin duoc la:4

```

Hình 28. Kết quả chương trình tìm kiếm nhị phân

Từ chương trình này, chúng ta rút ra được một số lưu ý như sau:

- ◊ Phần cơ bản của chương trình sẽ gồm một cấu trúc lặp (chưa xác định trước được số lần lặp);
- ◊ Cần ghi nhận được sự kiện tìm thấy, có thể dùng một biến logic *Tim_thay* để ghi nhận. Khi chưa tìm kiếm, tất nhiên phải khởi tạo biến này là *False*. Khi tìm thấy sẽ đổi giá trị của biến *Tim_thay* thành *True*. Điều này làm dễ dàng xác định điều kiện lặp.
- ◊ Điều kiện lặp là gì? Thể hiện sự kiện chưa tìm thấy hoặc không gian tìm kiếm chưa rỗng bằng biểu thức logic nào?
- ◊ Khi kết thúc lặp, giá trị của biến logic *Tim_thay* cho biết có tìm thấy hay không, bởi vậy sau cấu trúc lặp, dùng câu lệnh rẽ nhánh theo giá trị của *Tim_Thay* để thông báo kết quả.
- ◊ Trường hợp tìm thấy, cần đưa ra kết quả chi tiết hơn, cần thông báo chỉ số của phần tử có giá trị k . Khi tìm thấy, sự kiện này được ghi nhận ngay (ngay sau khi so sánh phần tử *giữa* được chọn với k), rồi điều kiện lặp được kiểm tra và

quá trình lặp dừng lại. Bởi vậy, lúc đó biến *Gima* cho biết chỉ số của phần tử cần tìm.

2. Kiểu mảng hai chiều

- *Mảng hai chiều* là bảng các phần tử cùng kiểu, là mảng một chiều mà mỗi phần tử của nó lại là một mảng một chiều.
- Với kiểu mảng hai chiều, các ngôn ngữ lập trình có các quy tắc, cách thức cho phép xác định:
 - Tên kiểu mảng hai chiều;
 - Số lượng phần tử của mỗi chiều;
 - Kiểu dữ liệu của phần tử;
 - Cách khai báo biến;
 - Cách tham chiếu đến phần tử.

Ví dụ, bảng cửu chương có thể được khai báo bằng Pascal như sau:

```
var B: array[1..9] of array[1..10] of integer;
```

hoặc có thể khai báo ngắn gọn hơn như sau:

```
var B: array[1..9, 1..10] of integer;
```

❖ Khai báo

Cách 1: Khai báo trực tiếp biến mảng hai chiều như sau:

```
var <tên biến mảng>: array[kiểu chỉ số dòng, kiểu chỉ số cột] of <kiểu phần tử>;
```

Cách 2: Khai báo gián tiếp biến mảng qua kiểu mảng hai chiều:

```
type <tên kiểu mảng>=array[kiểu chỉ số dòng, kiểu chỉ số cột] of <kiểu phần tử>;
```

```
var <tên biến mảng>: <tên kiểu mảng>;
```

• Ví dụ

Các khai báo sau đây là hợp lệ:

```
type
```

```
ArrayReal = array[-100..200, 100..200] of real;
```

```
ArrayBoolean = array[-n+1..n+1, n..2*n] of boolean;
```

```
var
```

```
ArrayInt: array[1..10, ..15] of integer;
```

```
ArrayLong: array[0..3*(n+1), 0..n] of longint;
```

Trong đó, n là hằng nguyên.

Tham chiếu tới phần tử của mảng hai chiều được xác định bởi tên mảng cùng với hai chỉ số được cách nhau bởi dấu phẩy và viết trong cặp dấu ngoặc [và].

Ví dụ

Tham chiếu tới phần tử ở dòng 5, cột 9 của biến mảng *ArrayInt* được viết như sau:
ArrayInt[5,9];

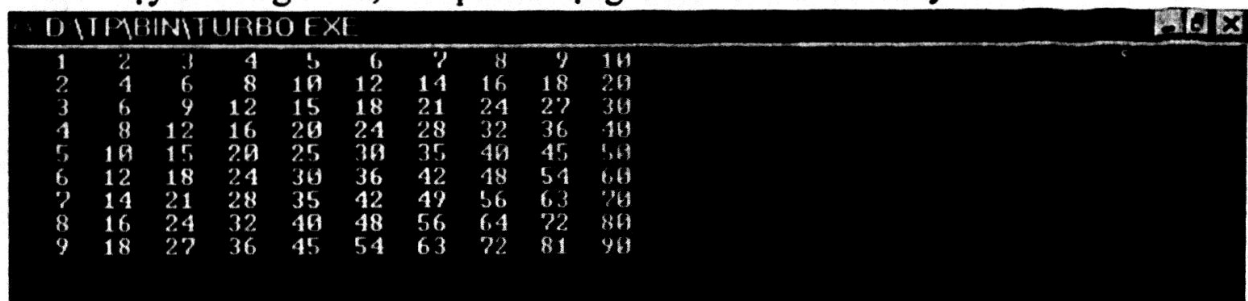
❖ Lưu ý:

- Giống như khi khai báo kiểu dữ liệu mảng một chiều, người lập trình cần phải xác định kiểu của các phần tử tạo nên mảng và kiểu chỉ số. Cách xác định kiểu chỉ số vẫn như đã biết ở kiểu mảng một chiều, chỉ khác là ở mảng hai chiều cần xác định hai chỉ số, hai chỉ số đó độc lập với nhau.
 - Giống như ở mảng một chiều, các thao tác nhập, xuất hay xử lý mỗi phần tử của mảng phải tuân theo quy định kiểu phần tử của mảng.
 - Việc thực hiện các thao tác nào đó (nhập, xuất hay xử lý) lần lượt trên các phần tử của mảng hai chiều thường gắn với hai câu lệnh *for-do* lồng nhau.
- Đối với ví dụ 1 về bài toán: “*Tính và đưa ra màn hình bảng cửu chương*”.

Chương trình tính và đưa ra màn hình bảng cửu chương:

```
program Bangcuuchuong;  
uses crt;  
var  
    B: array[1..9, 1..10] of integer;  
    {B: biến mảng hai chiều lưu bảng cửu chương}  
    i, j: integer;  
Begin  
    clrscr;  
    for i:=1 to 9 do  
        for j:=1 to 10 do  
            B[i,j]:= i*j;  
    for i:=1 to 9 do  
        begin  
            for j:=1 to 10 do write(B[i,j]:4);  
            writeln;  
        end;  
        readln  
    End.
```

Khi chạy chương trình, kết quả có dạng như hình 29 dưới đây:



1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90

Hình 29. Kết quả chương trình tính và đưa ra màn hình bảng cửu chương

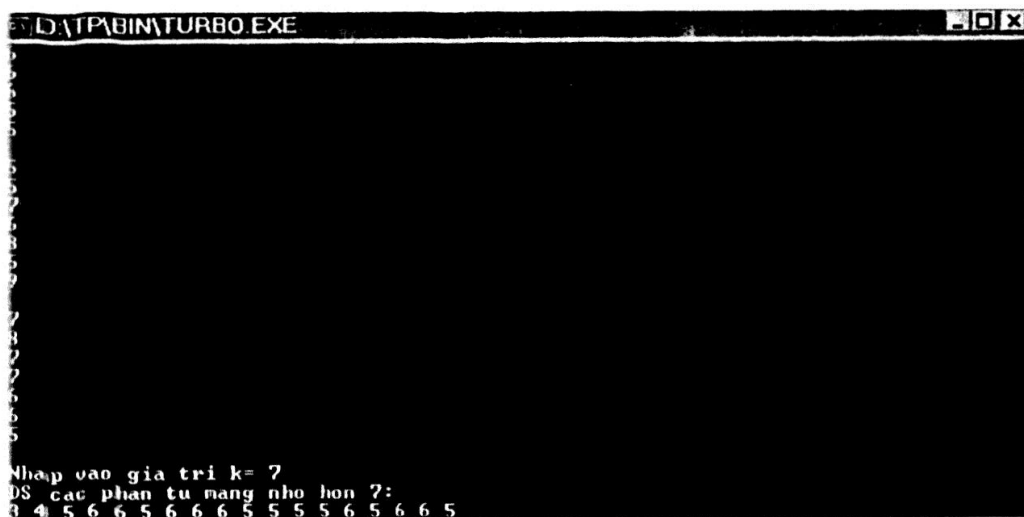
- Đối với ví dụ 2 về bài toán: “*Lập chương trình nhập vào từ bàn phím các phần tử của mảng hai chiều B gồm 5 dòng, 7 cột với các phần tử là các số nguyên*”

và số nguyên k . Sau đó, đưa ra màn hình các phần tử của mảng có giá trị nhỏ hơn k ".

➤ *Chương trình:*

```
program manghaichieu_1;
uses crt;
var b: array[1..5, 1..7] of integer;
    d, i, j, k: integer;
Begin
clrscr;
writeln('Nhap cac phan tu cua mang theo dong:');
for i:= 1 to 5 do
    begin
        for j:= 1 to 7 do read(b[i,j]);
        writeln;
    end;
write('Nhap vao gia tri k= '); readln(k);
d:=0;
writeln('DS cac phan tu mang nho hon ', k, ':');
for i:=1 to 5 do
    for j:= 1 to 7 do
        if b[i,j] < k then begin
            write(b[i,j], ' ');
            d:= d+1;
        end;
if d=0 then writeln('khong co phan tu nao nho hon', k);
readln
End.
```

Sau khi chạy chương trình, nhập dữ liệu với giá trị các phần tử của mảng và giá trị $k=7$ thì danh sách các phần tử nhỏ hơn 7 được in ra màn hình và kết quả của chương trình như hình 30 dưới đây:



Hình 30

Nhưng khi nhập vào một giá trị $k = 3$ nhỏ hơn các phần tử của mảng đã nhập vào thì chương trình sẽ in ra thông báo: “*không có phần tử nào nhỏ hơn 3*” (Hình 31)



The screenshot shows a Turbo Pascal IDE window titled "D:\TP\BIN\TURBO.EXE". The program has executed and produced the following output:

```

98
6
6
6
8
6
5
8
9
9
6
6
5
5
5
Nhập vào giá trị k= 3
DS các phần tử mảng nhỏ hơn 3:
không có phần tử nào nhỏ hơn 3

```

Hình 31. Kết quả chương trình với $k=3$ (nhỏ hơn các phần tử của mảng)

➤ Đối với chương trình trên, ta có thể thay đổi hình thức nhập các phần tử của mảng vào từ bàn phím. Khi đó, chương trình sẽ như sau:

```

program manghaichieu_2;
uses crt;
var b: array[1..5, 1..7] of integer;
    d, i, j, k: integer;
Begin
clrscr;
writeln('Nhập các phần tử của mảng theo dòng:');
for i:= 1 to 5 do
    begin
        for j:= 1 to 7 do
            begin
                write('B[' , i , ',' , j , ']=');
                read(b[i,j]);
            end;
        writeln;
    end;
write('Nhập vào giá trị k= '); readln(k);
d:=0;
writeln('DS các phần tử mảng nhỏ hơn ' , k , ':');
for i:=1 to 5 do
    for j:= 1 to 7 do
        if b[i,j] < k then begin
            write(b[i,j], ' ');
            d:= d+1;
        end;

```

```

if d=0 then writeln('khong co phan tu nao nho hon', k);
readln
End.

```

Kết quả chương trình có dạng như hình 32 dưới đây:

```

DATPABIN\TURBO.EXE
B[3,3]=65
B[3,4]=3
B[3,5]=5
B[3,6]=6
B[3,7]=7

B[4,1]=6
B[4,2]=8
B[4,3]=7
B[4,4]=6
B[4,5]=5
B[4,6]=5
B[4,7]=6

B[5,1]=5
B[5,2]=6
B[5,3]=7
B[5,4]=5
B[5,5]=6
B[5,6]=7
B[5,7]=5

Nhap vao gia tri k = 6
DS cac phan tu mang nho hon 6:
1 4 5 3 2 5 5 3 4 3 5 5 5 5 5 5

```

Hình 32



Bài tập và thực hành 3

1. Mục đích, yêu cầu

- Nâng cao kỹ năng sử dụng một số câu lệnh và một số kiểu dữ liệu thông qua việc tìm hiểu, chạy thử các chương trình có sẵn.
- Biết giải một số bài toán tính toán, tìm kiếm đơn giản trên máy tính.

2. Nội dung

Bài 1. Tạo mảng A gồm n ($n \leq 100$) số nguyên, mỗi số có giá trị tuyệt đối không vượt quá 300. Tính tổng các phần tử của mảng là bội số của một số nguyên dương k cho trước.

a) Hãy tìm hiểu và chạy thử chương trình sau đây:

```

program Sum1;
uses crt;
const nmax=100;
type MyArray= array[1..nmax] of integer;
var A: MyArray;
    s, n, i, k: integer;

```

```

Begin
  clrscr; randomize;
  write('Nhap n= ');
  readln(n); {Tao ngau nhien mang gom n so nguyen}
  for i:=1 to n do A[i]:= random(300)-random(300);
  for i:=1 to n do write(A[i]:5); {in ra mang vua tao}
  writeln;
  write('Nhap k = ');
  readln(k);
  s:=0;
  for i:= 1 to n do
    if A[i] mod k = 0 then s:= s+ A[i];
  writeln('Tong can tinh la: ',s);
  readln
end.

```

Chú ý: Hàm chuẩn *random(n)* cho giá trị là số nguyên ngẫu nhiên trong khoảng từ 0 đến $n-1$, còn thủ tục *randomize* khởi tạo cơ chế sinh số ngẫu nhiên.

b) Hãy đưa các câu lệnh sau đây vào những vị trí cần thiết nhằm sửa đổi chương trình trong câu a) để có được chương trình đưa ra các số dương và các số âm trong mảng.

```

posi, neg: integer;
posi:= 0; neg:= 0;
if A[i]>0 then posi:= posi + 1
  else if A[i]<0 then neg:= neg + 1;
writeln(posi:4, neg:4);

```

Bài 2: Viết chương trình tìm phần tử có giá trị lớn nhất của mảng và đưa ra màn hình chỉ số và giá trị của phần tử tìm được. Nếu có nhiều phần tử có cùng giá trị lớn nhất thì đưa ra màn hình có chỉ số nhỏ nhất.

a) Hãy tìm hiểu chương trình sau đây:

```

Program MaxElement;
const Nmax= 100;
type Myarray = array[1..Nmax] of integer;
var A: MyArray;
    n, i, j: integer;
Begin
  write('Nhap so luong phan tu cua day so, N= ');
  readln(N);
  for i:= 1 to N do
    begin
      write('Phan tu thu ', i, '=');
      readln(A[i]);
    end;
  j:= 1;
  for i:= 1 to n do if A[i]>A[j] then j:= i;

```

```

    write('Chi so: ',j,' Gia tri: ',A[j]: 4);
    readln
End.

```

b) Chỉnh sửa chương trình trên để đưa ra chỉ số của các phần tử có cùng giá trị lớn nhất.

Hướng dẫn làm bài tập và thực hành 3

1. Mục đích, yêu cầu

- Nâng cao kỹ năng sử dụng một số câu lệnh và một số kiểu dữ liệu thông qua việc tìm hiểu, chạy thử các chương trình có sẵn.
- Biết giải một số bài toán tính toán, tìm kiếm đơn giản trên máy tính.

2. Nội dung

Bài 1: Tạo mảng A gồm n số nguyên ($n \leq 100$) theo yêu cầu của đề bài ở trên thì tổng các phần tử (giá trị tuyệt đối của các phần tử không vượt quá 300) của mảng là bội của một số nguyên k cho trước nghĩa là ta cần tính tổng của các phần tử mà các phần tử này chia hết cho số nguyên k .

a) Sau khi nhập chương trình và cho chương trình chạy và nhập dữ liệu, ta có kết quả như hình 33 dưới đây:

```

D:\TP\BIN\TURBO.EXE
Nhap n=50
68 193 -31 28 -127 -1 -166 -137 59 123 -116 47 45 -157 -120 40
42 -260 -37 -19 28 96 122 175 -3 -39 25 43 -16 -100 139 -140
-130 13 212 -141 -148 124 -57 15 99 58 16 -103 -123 -21 -88 -191
46 215
Nhap k = 9
Tổng cần tính là:144

```

Hình 33

Trong hình ở trên, số phần tử cần nhập vào là 50. Khi đó, mảng được tạo ra một cách ngẫu nhiên gồm 50 số nguyên có giá trị tuyệt đối không vượt quá 300. Tiếp đến ta nhập vào số $k = 9$. Ta nhận thấy rằng, trong 50 số được tạo ra thì chỉ các số 45 và 99 là chia hết cho 9 và tổng của chúng bằng 144.

Tương tự như trên, nếu ta nhập $n = 99$, $k = 10$ thì kết quả cho như ở hình 34 dưới đây:

```

D:\TP\BIN\TURBO.EXE
Nhap n=99
-94 -63 33 267 -238 66 -52 -10 62 -187 134 79 62 -97 -97 38
-95 244 -37 -214 -2 4 21 65 -134 -144 88 -78 83 102 -262 276
21 -229 152 45 -116 -12 -69 163 -39 -72 -6 163 49 34 136 -66
16 -84 177 -238 126 -78 -26 -166 -110 -212 113 -172 -3 -238 -24 72
-129 -3 84 76 -204 26 11 146 -30 113 -26 69 -163 -33 105 -31
36 -49 87 46 -64 245 73 88 85 97 -123 251 -58 -149 40 119
-123 -267 -30
Nhap k = 10
Tổng cần tính là:-140

```

Hình 34. Kết quả chương trình với $n=99$, $k=10$

Chúng ta cũng dễ dàng nhận ra rằng, các số: -10, -110, -30, 40 và -30 là bội của 10 và tổng của chúng bằng -140.

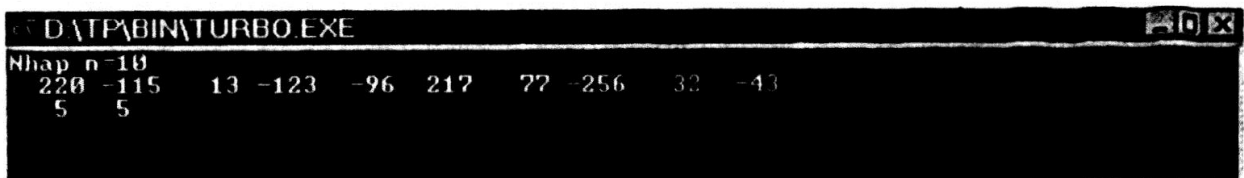
b) Sau khi chương trình ở câu a) đã chạy và cho kết quả tốt, ta thực hiện lệnh **File/save** để ghi lại chương trình vào máy tính, chẳng hạn với tên tệp **btth3_1a.doc**. Để tiến hành giải bài toán ở câu b) thì ta thực hiện lệnh **File/Save as** để ghi chương trình sang một tệp khác. Ví dụ, với tên **btth3_1b.doc**. Sau đó, đưa các câu lệnh

```
posi, neg: integer;
posi:= 0; neg:= 0;
if A[i]>0 then posi:= posi + 1
    else if A[i]<0 then neg:= neg + 1;
writeln(posi:4, neg:4);
```

vào chương trình của câu a). Chương trình để in ra số các số dương và số các số âm trong mảng *A* như sau:

```
program Sum1b;
uses crt;
const nmax=100;
type Myarray= array[1..nmax] of integer;
var A: MyArray;
    n, i: integer;
    posi, neg: integer;
Begin
    clrscr; randomize;
    write('Nhap n=');
    readln(n); {Tao ngau nhien mang gom n so nguyen}
    for i:=1 to n do A[i]:=random(300)-random(300);
    for i:=1 to n do write(A[i]:5); {in ra mang vua tao}
    writeln;
    posi:= 0; neg:= 0;
    for i:= 1 to n do
        if A[i]>0 then posi:= posi + 1
        else if A[i]<0 then neg:= neg + 1;
    writeln(posi:4, neg:4);
    readln
End.
```

Khi chạy chương trình, nhập số phần tử của mảng, ví dụ, $n = 10$ thì kết quả của chương trình sẽ như hình 35 sau đây:



```
D:\TP\BIN\TURBO.EXE
Nhap n=10
220 -115 13 -123 -96 217 77 -256 32 -43
5 5
```

Hình 35. Kết quả chương trình in ra số các số dương, số các số âm trong mảng *A*

Với kết quả như ở hình 35 thì người sử dụng nhiều khi rất khó nhìn thấy số các số dương và số các số âm. Bởi vậy, chúng ta cần đưa vào lệnh in dòng chú thích:

```
writeln('So cac so duong la : ', posi:4);
writeln('So cac so am la      : ', neg:4);
```

Khi đó, chương trình in ra số các số dương và số các số âm trong mảng *A* là:

```
program Sum1b2;
uses crt;
const nmax=100;
type Myarray= array[1..Nmax] of integer;
var A: MyArray;
    n, i: integer;
    posi, neg: integer;
Begin
  clrscr; randomize;
  write('Nhap n=');
  readln(n); {Tao ngau nhien mang gom n so nguyen}
  for i:=1 to n do A[i]:=random(300)-random(300);
  for i:=1 to n do write(A[i]:5); {in ra mang vua tao}
  writeln;
  posi:= 0; neg:= 0;
  for i:= 1 to n do
    if A[i]>0 then posi:= posi + 1
    else if A[i]<0 then neg:= neg + 1;
  writeln('So cac so duong la: ', posi:4);
  writeln('So cac so am la      : ', neg:4);
  readln
End
```

Khi chạy chương trình, nhập số phần tử của mảng, ví dụ, $n = 50$ thì chương trình sẽ đưa ra thông báo:

```
So cac so duong la: 24
So cac so am la: 26
```

Kết quả của chương trình sẽ như hình 36 dưới đây:

The screenshot shows a window titled 'LAT P\BIN\TURBO.EXE'. The output text is as follows:

```
Nhap n= 50
-29  54  106  41 -177 -131  -68   47  -83 -141  -28  242  -16  212  -206  -43
 23  226  -46 -134 -114  -38 -124  184   52  100   21 -141   75  130  223   22
-16   72  -98   23   11   -9 -146 -167   68  203  -87 -166   98   28  -11 -233
  1  252
So cac so duong la: 24
So cac so am la   : 26
```

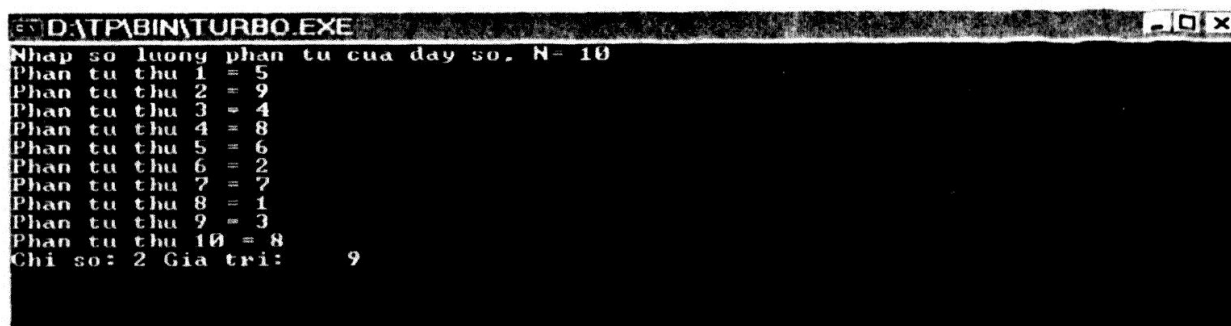
Hình 36. Kết quả chương trình in ra số các số dương, số các số âm trong mảng *A*

Bài 2: Tìm phần tử có giá trị lớn nhất của mảng và đưa ra màn hình chỉ số và giá trị của phần tử tìm được. Khi có nhiều phần tử có cùng giá trị lớn nhất thì đưa ra phần tử có chỉ số nhỏ nhất.

a) Sau khi nhập chương trình và cho chương trình chạy và nhập số lượng phần tử của dãy số với $n = 10$ với giá trị các phần tử như sau:

Phần tử thứ nhất= 5;
Phần tử thứ hai= 9;
Phần tử thứ ba= 4;
Phần tử thứ tư= 8;
Phần tử thứ năm= 6;
Phần tử thứ sáu= 2;
Phần tử thứ bảy= 7;
Phần tử thứ tám= 1;
Phần tử thứ chín= 3;
Phần tử thứ mười= 8;

thì chương trình in ra thông báo: "Chỉ số: 2 Giá trị: 9" nghĩa là phần tử ở chỉ số 2 có giá trị 9 là phần tử lớn nhất mảng. Khi đó, ta có kết quả như hình 37 dưới đây:



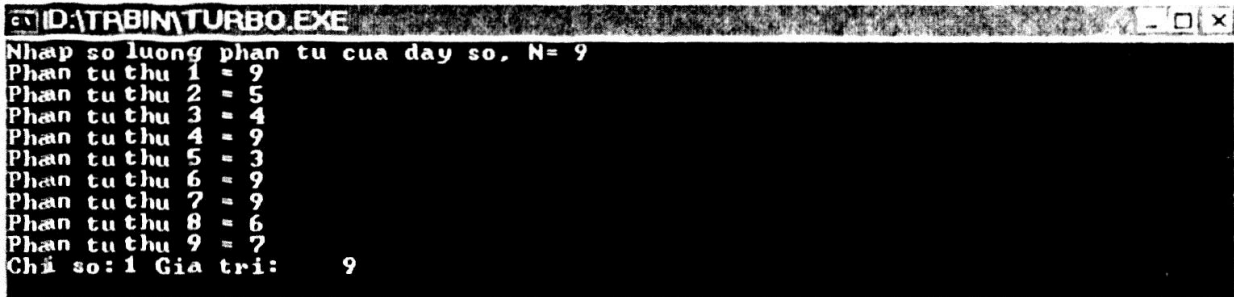
```
D:\TP\BIN\TURBO.EXE
Nhập số lượng phần tử của dãy số, N= 10
Phần tử thứ 1 = 5
Phần tử thứ 2 = 9
Phần tử thứ 3 = 4
Phần tử thứ 4 = 8
Phần tử thứ 5 = 6
Phần tử thứ 6 = 2
Phần tử thứ 7 = 7
Phần tử thứ 8 = 1
Phần tử thứ 9 = 3
Phần tử thứ 10 = 8
Chỉ số: 2 Giá trị: 9
```

Hình 37. Kết quả của chương trình với $n = 10$

Nhưng khi nhập các phần tử vào mảng mà có nhiều phần tử có cùng giá trị lớn nhất thì kết quả của chương trình sẽ in ra chỉ số nhỏ nhất của phần tử lớn nhất. Chẳng hạn, nhập vào 9 phần tử với giá trị các phần tử được nhập vào mảng như sau:

Phần tử thứ nhất= 9;
Phần tử thứ hai= 5;
Phần tử thứ ba= 4;
Phần tử thứ tư= 9;
Phần tử thứ năm= 3;
Phần tử thứ sáu= 9;
Phần tử thứ bảy= 9;
Phần tử thứ tám= 6;
Phần tử thứ chín= 7;

Khi đó chương trình sẽ đưa ra thông báo: "Chỉ số: 1 Gia trị: 9" như hình 38 dưới đây:



Hình 38. Chương trình với nhiều phần tử có cùng giá trị lớn nhất

Với kết quả của chương trình hiện ra như ở hình 38 thì chỉ số nhỏ nhất của phần tử lớn nhất (bằng 9) là 1.

Lưu ý: Trong chương trình chúng ta nên đưa vào khai báo chuẩn *uses crt*; và lệnh xóa màn hình *clrscr*; để mỗi lần chạy chương trình các kết quả trước đó bị xóa:

Khi đó chương trình sẽ là:

```
Program MaxElement;  
uses crt;  
const Nmax= 100;  
type Myarray = array[1..Nmax] of integer;  
var A: MyArray;  
    i, j: integer;  
Begin  
Clrscr;  
    write('Nhap so luong phan tu cua day so, N= ');  
    readln(N);  
    for i:= 1 to N do  
        begin  
            write('Phan tu thu ', i, '=');  
            readln(A[i]);  
        end;  
    j:= 1;  
    for i:= 1 to n do if A[i]>A[j] then j:= i;  
    write('Chỉ số: ', j, ' Gia trị: ', A[j]: 4);  
    readln  
End.
```

b) Sau khi chương trình ở câu a) đã chạy và cho kết quả tốt, ta thực hiện lệnh **File/save** để ghi lại chương trình vào máy tính, chẳng hạn với tên tệp **btth3_2a.doc**. Để tiến hành giải bài toán ở câu b) thì ta thực hiện lệnh **File/Save as** để ghi chương trình sang một tệp khác. Ví dụ, với tên **btth3_2b.doc**. Sau đó, ta chỉnh sửa chương trình để đưa ra chỉ số của các phần tử có cùng giá trị lớn nhất, bằng cách ta đưa vào chương trình biến mảng *dem*. Chương trình như sau:

```

Program MaxElement2;
const Nmax= 100;
type Myarray = array[1..Nmax] of integer;
var A: Myarray;
    dem: array[1..Nmax] of integer;
    n, k, i, j: integer;
Begin
    write('Nhap so luong phan tu cua day so, N= ');
    readln(N);
    for i:= 1 to n do
        begin
            write('Phan tu thu ', i, ' = ');
            readln(A[i]);
        end;
    j:= 1; k:= 0;
    for i:= 2 to n do if A[i] > A[j] then j:= i;
        dem[k]:= j;
        for i:= 1 to n do if a[i] = a[j] then
            begin
                k:= k + 1;
                dem[k]:= i;
            end;
        writeln('Gia tri lon nhat la: ', A[j]:4);
        writeln('Chi so ung voi gia tri lon nhat la: ');
        for i:= 1 to k do write(' ', dem[i]);
        writeln;
    readln;
End.

```

Khi cho chương trình chạy và nhập số lượng phần tử của dãy số với $n = 5$. Giả sử, giá trị các phần tử được nhập vào như sau:

Phần tử thứ nhất= 4;
 Phần tử thứ hai= 7;
 Phần tử thứ ba= 9;
 Phần tử thứ tư= 5;
 Phần tử thứ năm= 9;

Khi đó chương trình sẽ đưa ra thông báo:

Gia tri lon nhat la: 9
 Chi so ung voi gia tri lon nhat la:
 3 5

Kết quả của chương trình như hình 39 dưới đây:

```

D:\TP\BINTURBO.EXE
Turbo Pascal Version 7.0 Copyright (c) 1983,92 Borland International
Nhap: so luong phan tu cua day so, N= 5
Phan tu thu 1 = 4
Phan tu thu 2 = 7
Phan tu thu 3 = 9
Phan tu thu 4 = 5
Phan tu thu 5 = 9
Gia tri lon nhat la: 9
Chi so ung voi gia tri lon nhat la:
3 5

```

Hình 39



Bài tập và thực hành 4

1. Mục đích, yêu cầu

- *Biết nhận xét, phân tích đề xuất thuật toán giải bài toán sao cho chương trình chạy nhanh hơn;*
- *Làm quen với dữ liệu có cấu trúc và bài toán sắp xếp.*

2. Nội dung

Bài 1

- a) Hãy tìm hiểu và chạy thử chương trình thực hiện thuật toán sắp xếp dãy số nguyên bằng trao đổi với các giá trị khác nhau của n dưới đây. Qua đó, nhận xét về thời gian chạy của chương trình.

(*chương trình giải bài toán sắp xếp dãy số*)

```

uses crt;
const Nmax= 250;
type ArrInt = Array[1..Nmax] of integer;
var n, i, j, y, t: integer;
    A: ArrInt;
begin
  clrscr;
  randomize;
  write('Nhap n= ');
  readln(n);
  {Tạo ngẫu nhiên mảng gồm n số nguyên}
  for i:= 1 to n do A[i]:= random(300)-random(300);
  for i:= 1 to n do write(A[i]:5); {in mảng vừa tạo}
  writeln;
  for j:=N downto 2 do
    for i:= 1 to j-1 do
      if A[i]>A[i+1] then

```

```

begin (*Trao doi A[i] va A[i+1]*)
    t:= A[i];
    A[i]:= A[i+1];
    A[i+1]:= t;
end;
writeln('Day so duoc sap xep:');
for i:= 1 to n do
    writeln(A[i]:7);
writeln;
readln
End.

```

- b) Khai báo thêm biến nguyên *dem* và bổ sung vào chương trình những câu lệnh cần thiết để biến *dem* tính số lần thực hiện trao đổi trong thuật toán. Đưa kết quả tìm được ra màn hình.

Bài 2:

- a) Hãy đọc và tìm hiểu những phân tích để viết chương trình giải bài toán:

Cho mảng A gồm n phần tử. Hãy viết chương trình tạo mảng B[1..n], trong đó B[i] là tổng của i phần tử đầu tiên của A.

Thoạt đầu có thể viết chương trình sau để giải bài toán này:

```

program subsum1;
uses crt;
const nmax=100;
type Myarray= array[1..nmax] of integer;
var A, B: Myarray;
    n, i, j: integer;
Begin
clrscr;
randomize;
write(' Nhap n:');
readln(n);
{Tao ngau nhien mang gom n so nguyen}
for i:= 1 to n do A[i]:= random(300)-random(300);
for i:= 1 to n do write(A[i]:5);
    writeln;
    {Bat dau}
    for i:= 1 to n do
        begin
            B[i]:= 0;
            for j:= 1 to i do B[i]:= B[i] + A[j];
        end;
    {ket thuc}
    for i:= 1 to n do write(B[i]:6);
readln
End.

```


Để ý rằng ta có các hệ thức sau:

$$E[1] = A[1]$$

$$E[i] = B[i-1] + A[i], \quad 1 < i \leq n.$$

Phân tích đó cho phép thay đoạn chương trình từ chú thích *{bat dau}* đến *{ket thuc}* bởi hai lệnh sau:

$B[1] := A[1];$

for $i := 2$ to n do $B[i] := B[i-1] + A[i];$

Với hai lệnh này, máy chỉ phải thực hiện $n - 1$ phép cộng, trong khi với đoạn chương trình trên máy phải thực hiện $\frac{n(n+1)}{2}$ phép cộng.

Nhờ việc phân tích ta có thể tiết kiệm được một lượng tính toán đáng kể.

Tuy tốc độ tính toán của máy tính nhanh nhưng có giới hạn. Do đó, trong khi viết chương trình, ta nên tìm cách viết sao cho chương trình thực hiện càng ít phép toán càng tốt.

Hướng dẫn làm bài tập và thực hành 4

1. Mục đích, yêu cầu

- *Biết nhận xét, phân tích đề xuất thuật toán giải bài toán sao cho chương trình chạy nhanh hơn;*
- *Làm quen với dữ liệu có cấu trúc và bài toán sắp xếp.*

2. Nội dung

Bài 1:

a) Sau khi nhập chương trình, cho chương trình chạy và nhập số phần tử của dãy, chẳng hạn với $n = 10$, thì giá trị các phần tử của mảng ngẫu nhiên được tạo ra như sau:

Phần tử thứ nhất= 100;

Phần tử thứ hai= -122;

Phần tử thứ ba= 22;

Phần tử thứ tư= -40;

Phần tử thứ năm= 71;

Phần tử thứ sáu= 31;

Phần tử thứ bảy= -141;

Phần tử thứ tám= 29;

Phần tử thứ chín= -99;

Phần tử thứ mười= 162;

thì chương trình in ra thông báo:

Day so duoc sap xep:

-141
-122
-99
-40
22
29
31
71
100
162

Kết quả chương trình như hình 40 dưới đây:

```
D:\TP\BIN\TURBO EXE
Nhap n= 10
100 -122 22 -40 71 31 -141 29 -99 162
Day so duoc sap xep:
-141
-122
-99
-40
22
29
31
71
100
162
```

Hình 40. Kết quả chương trình với $n = 10$

Ta nhận thấy rằng, mảng được sắp xếp theo trình tự từ số bé nhất đến số lớn nhất của mảng. Trong trường hợp với $n = 10$ phần tử như đã nói ở trên thì mảng được tạo ngẫu nhiên và sắp xếp từ số bé nhất -141 đến số lớn nhất 162.

Tương tự, với $n = 15$ thì kết quả của chương trình sẽ như hình 41 dưới đây:

```
D:\TP\BIN\TURBO EXE
Nhap n= 15
-102 -33 -119 120 -54 -52 -212 288 -115 78 106 99 192 -168 77
Day so duoc sap xep:
-288
-212
-168
-119
-115
-102
-52
-54
33
77
78
99
106
120
192
```

Hình 41. Kết quả chương trình với $n = 15$

b) Sau khi chương trình ở câu a) đã chạy và cho kết quả tốt, ta thực hiện lệnh **File/save** để ghi lại chương trình vào máy tính, chẳng hạn với tên tệp **btth4_1a.doc**. Để tiến hành giải bài toán ở câu b) thì ta thực hiện lệnh **File/Save as** để ghi chương trình sang một tệp khác. Chẳng hạn, với tên **btth4_1b.doc**.

Khi đưa biến nguyên *dem* vào chương trình và bổ sung vào chương trình “Giải bài toán sắp xếp dãy số” các lệnh:

```
dem:= 0;  
dem:= dem + 1;  
write('So lan trao doi la: ',dem);
```

để thực hiện việc *tính số lần trao đổi trong thuật toán* thì chương trình được chỉnh lại như sau:

```
uses crt;  
const Nmax = 250;  
type ArrInt = Array[1..Nmax] of integer;  
var n, i, j, y, t, dem, : integer;  
    A: ArrInt;  
Begin  
clrscr;  
randomize;  
write('Nhap n= ');  
readln(n);  
{Tao nau nhien mang gom n so nguyen}  
for i:= 1 to n do A[i]:= random(300)-random(300);  
for i:= 1 to n do write(A[i]:5); {in mang vua tao}  
writeln;  
dem:=0;  
for j:=N downto 2 do  
    for i:= 1 to j-1 do  
        if A[i]>A[i+1] then  
            begin (*Trao doi A[i] va A[i+1]*)  
                t:= A[i];  
                A[i]:= A[i+1];  
                A[i+1]:= t;  
                dem:= dem + 1;  
            end;  
    writeln('Day so duoc sap xep:');  
    for i:= 1 to n do  
        writeln(A[i]:7);  
    write('So lan trao doi la: ',dem);  
    writeln;  
    readln  
End.
```

Khi chạy chương trình với $n=8$ thì kết quả của chương trình như hình 42 dưới đây:

```

D:\TP\BIN\TURBO EXE
Nhap n= 8
-47 9 169 -190 -20 -27 -80 -33
Day so duoc sap xep:
-190
-80
-47
-33
-27
-20
9
169
Số lần trao đổi là: 17

```

Hình 42. Kết quả chương trình tính số lần trao đổi trong thuật toán

Bài 2:

a) Khi chạy chương trình tạo mảng $B[1..n]$, $B[i]$ là tổng của i phần tử đầu tiên của mảng A - gồm n phần tử và nhập vào số phần tử của mảng thì ta có được kết quả của chương trình.

- Với $n=10$ thì kết quả của chương trình như hình 43 dưới đây:

```

D:\TP\BIN\TURBO EXE
Nhap n:10
-118 6 -42 -22 -37 11 243 19 90 116
-118 -112 -154 -176 -213 -202 41 60 150 266

```

Hình 43

Diễn giải: Từ kết quả của chương trình đã cho ở trên, ta nhận thấy rằng: với $n=10$, mảng A được tạo ngẫu nhiên là: -118 6 -42 -22 -37 11 243 19 90 116. Khi đó, các phần tử của mảng B sẽ là như sau:

- phần tử đầu tiên của mảng B là -118;
- phần tử thứ hai của mảng B là: $-118 + 6 = -112$;
- phần tử thứ ba của mảng B là: $-112 + (-42) = -154$;
- phần tử thứ tư của mảng B là: $-154 + (-22) = -176$;
- phần tử thứ năm của mảng B là: $-176 + (-37) = -213$;
- phần tử thứ sáu của mảng B là: $-213 + 11 = -202$;
- phần tử thứ bảy của mảng B là: $-202 + 243 = 41$;
- phần tử thứ tám của mảng B là: $41 + 19 = 60$;
- phần tử thứ chín của mảng B là: $60 + 90 = 150$;
- phần tử thứ mười của mảng B là: $150 + 116 = 266$.

Như vậy, các phần tử của mảng B là -118 -112 -154 -176 -213 -202 41 60 150 226.

- Với $n=15$ thì kết quả của chương trình như hình 44 dưới đây:

```

D:\TP\BIN\TURBO.EXE
Nhap n:15
-81 -49 129 98 16 112 180 -173 70 -123 -29 -96 -173 26 25
-18 -137 -8 90 106 218 398 225 295 172 143 47 -126
-100 -75

```

Hình 44

b) Khi thay đoạn chương trình sau ở câu a):

```

for i:= 1 to n do
begin
  B[i]:= 0;
  for j:=1 to i do B[i]:= B[i]+A[j];
end;

```

bởi hai lệnh:

```

B[1]:= A[1];
for i:= 2 to n do B[i] := B[i-1] + A[i];

```

thì chương trình tạo mảng $B[1..n]$, trong đó $B[i]$ là tổng của i phần tử đầu tiên của mảng A sẽ là như sau:

```

program subsum2;
uses crt;
const nmax= 100;
type Myarray= array[1..nmax] of integer;
var A, B: Myarray;
    n, i, j: integer;
Begin
  clrscr;
  randomize;
  write(' Nhap n:');
  readln(n);
  {Tao ngau nhien mang gom n so nguyen}
  for i:= 1 to n do A[i]:= random(300)- random(300);
  for i:= 1 to n do write(A[i]:5);
  writeln;
  B[1]:= A[1];
  for i:= 2 to n do B[i]:= B[i-1] + A[i];
  for i:= 1 to n do write(B[i]:6);
  readln
End.

```

➤ Khi chạy chương trình vừa hiệu chỉnh này, với số phần tử của mảng A bằng 9 thì kết quả của chương trình sẽ như hình 45 dưới đây:

```

D:\TP\BIN\TURBO.EXE
Nhap n 5
81 38 -57 68 -57
81 43 -14 54 -3

```

Hình 45

Diễn giải:

Với hai lệnh:

```
B[1] := A[1];  
for i := 2 to n do B[i] := B[i-1] + A[i];
```

thì kết quả chương trình sẽ không thay đổi so với chương trình ban đầu nghĩa là:

mảng ban đầu A ngẫu nhiên được tạo: 81 -38 -57 68 -57 thì với lệnh

$B[1] := A[1];$ thì phần tử đầu tiên của mảng B là 81;

Tiếp đến, khi $i = 2$ thì phần tử thứ hai của mảng B là: $81 + (-38) = 43$;

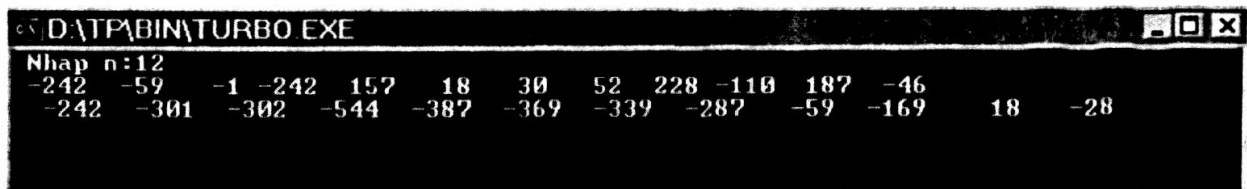
Tiếp đến, khi $i = 3$ thì phần tử thứ ba của mảng B là: $43 + (-57) = -14$;

Tiếp đến, khi $i = 4$ thì phần tử thứ tư của mảng B là: $-14 + 68 = 54$;

Tiếp đến, khi $i = 5$ thì phần tử thứ năm của mảng B là: $54 + (-57) = -3$.

Như vậy, mảng B được tạo ra là: 81 43 -14 54 -3.

- Với số phần tử của mảng A bằng 20 thì kết quả của chương trình sẽ như hình 46 dưới đây:



Hình 46. Kết quả chương trình với $n=12$

Qua hai chương trình đã nêu ở trên, ta nhận thấy rằng chương trình thứ hai tiết kiệm được một số phép toán. Cho nên, nó sẽ tiết kiệm được thời gian chạy máy, tiết kiệm được bộ nhớ của máy tính.

§12. KIỂU XÂU

A. TÓM TẮT LÝ THUYẾT

- *Dữ liệu kiểu xâu* là dãy các kí tự. Ví dụ: 'Ha noi';
- *Một xâu* là một dãy các kí tự (trong bảng mã *ASCII*), có thể coi xâu như một mảng một chiều mà mỗi phần tử là *một kí tự*. Số lượng kí tự trong một xâu được gọi là *độ dài của xâu*. Xâu có độ dài bằng 0 là xâu rỗng.
- Các ngôn ngữ lập trình đều có quy tắc, cách thức cho phép xác định:
 - Tên kiểu xâu;
 - Cách khai báo biến kiểu xâu;
 - Số lượng kí tự của xâu;
 - Các thao tác với xâu;
 - Cách tham chiếu tới phần tử xâu.

- Biểu thức gồm các toán hạng là *biến xâu*, biến kí tự hoặc hằng xâu được gọi là *biểu thức xâu*.

1. Khai báo

- Biến kiểu xâu có thể khai báo như sau:

`var <tên biến>: string[độ dài lớn nhất của xâu];`

Lưu ý: Độ dài lớn nhất của xâu ≤ 255 .

Ví dụ

`var Hoten: string[26];`

- Trong mô tả xâu có thể bỏ qua phần khai báo độ dài, chẳng hạn:

`var chugiai: string;`

Khi đó, độ dài lớn nhất của xâu sẽ nhận giá trị mặc định là 255.

2. Các thao tác xử lí xâu

a) **Phép ghép xâu** được dùng để ghép nhiều xâu thành một (kể cả đối với các hằng và biến xâu).

Ví dụ: 'Nghe' + ' An'. Kết quả: Nghe An

b) **Các phép so sánh:** (=), (<>), (<), (>), (<=), (>=) có thứ tự ưu tiên thực hiện thấp hơn phép ghép xâu và thực hiện việc so sánh hai xâu theo quy tắc sau:

- Xâu *A* là lớn hơn xâu *B* nếu kí tự đầu tiên khác nhau giữa chúng kể từ trái sang trong xâu *A* có mã *ASCII* lớn hơn.
- Nếu *A* và *B* là các xâu có độ dài khác nhau và *A* là đoạn đầu của *B* thì *A* nhỏ hơn *B*.

Ví dụ: 'Que huong' < 'Que huong toi'

- Hai xâu được coi là bằng nhau nếu như chúng giống nhau hoàn toàn.

Ví dụ: 'Ha noi' = 'Ha noi'

c) Thủ tục *delete(st, vt, n)* thực hiện việc xóa *n* kí tự của biến xâu *st* bắt đầu từ vị trí *vt*.

Ví dụ: `st='abcdef';` thao tác `delete(st, 4, 2);` cho kết quả 'abcd'

d) Thủ tục *insert(s1, s2, vt)* chèn xâu *s1* vào biến xâu *s2*, bắt đầu ở vị trí *vt*.

Ví dụ: `s1='PC'; s2='IBM486';` thao tác `insert(s1, s2, 4);` cho kết quả 'IBMPC486'

e) Hàm *copy(S, vt, N)* tạo xâu gồm *N* kí tự liên tiếp bắt đầu từ vị trí *vt* của xâu *S*.

Ví dụ: `S='Bai hoc thu 9';` biểu thức `copy(S, 9, 5);` cho kết quả 'thu 9'

f) Hàm *length(s)* cho giá trị là độ dài xâu *s*.

Ví dụ: `S='Tin hoc'` thì biểu thức `length(S)` có độ dài là 7.

g) Hàm $pos(s1, s2)$ cho vị trí xuất hiện đầu tiên của chuỗi $s1$ trong chuỗi $s2$.

Ví dụ: $s2 = 'abcdef'$ thì biểu thức $pos('cd', s2)$ cho kết quả 3.

h) Hàm $upcase(ch)$ cho chữ cái in hoa ứng với chữ cái trong ch .

Ví dụ: 'd' thì biểu thức $upcase(ch)$ cho kết quả 'D'

Lưu ý:

- Chuỗi được tạo thành bởi các ký tự, trong đó có thể có *dấu cách*. Dấu cách thể hiện trong các văn bản là phần trống ngăn cách giữa hai từ viết liên tiếp. Ký tự này được gõ bằng phím dài nhất trên bàn phím (**Space Bar**);
- Trong chương trình, khi viết một chuỗi ký tự, ta phải viết chuỗi đó giữa hai dấu nháy đơn. Nhưng khi nhập từ bàn phím giá trị một chuỗi, ta chỉ gõ các ký tự thuộc chuỗi đó (rồi nhấn phím **Enter**).
- Chuỗi chỉ gồm một dấu cách được viết là ' '. Để viết chuỗi rỗng ta viết hai dấu nháy đơn liên nhau.
- Khi so sánh hai chuỗi, chuỗi có độ dài nhỏ hơn có thể là chuỗi lớn hơn (>), ví dụ:
'Anh' < 'Ba'
- Khi sử dụng lệnh gán, ta có thể gán trị là một ký tự cho một biến chuỗi ký tự nhưng việc gán trị là một chuỗi ký tự cho một biến kiểu ký tự là không hợp lệ dù chuỗi đó có độ dài bằng 1.
- Tham số của các hàm và thủ tục chuẩn phải hợp lý, chẳng hạn không thể dùng $Insert(s1, s2, 10)$ khi $length(s2) < 10$.

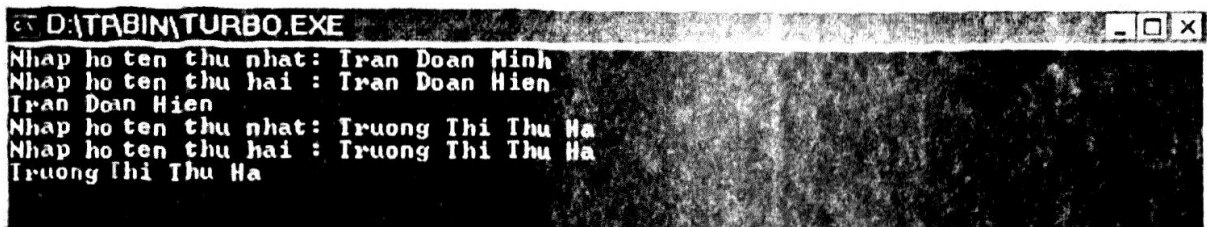
3. Một số ví dụ

Ví dụ 1: Bài toán so sánh hai chuỗi: nhập vào họ tên của hai người vào hai biến chuỗi và đưa ra màn hình chuỗi dài hơn, nếu bằng nhau thì đưa ra chuỗi nhập sau.

Chương trình:

```
Program vidu1;  
var  
    a, b: string;  
begin  
    write('Nhap ho ten thu nhat: '); readln(a);  
    write('Nhap ho ten thu hai : '); readln(b);  
    if length(a) > length(b) then write(a) else write(b);  
    readln  
end.
```

Khi chạy chương trình, nhập họ tên của hai người: Tran Doan Minh và Tran Doan Hien, thì kết quả của chương trình cho như hình 47 dưới đây:



Hình 47. Kết quả chương trình so sánh 2 xâu

Ví dụ 2: Bài toán kiểm tra hai xâu kí tự: “nhập vào hai xâu từ bàn phím và kiểm tra kí tự đầu tiên của xâu thứ nhất có trùng với kí tự cuối cùng của xâu thứ hai hay không?”

Chương trình:

```
program vidu2;
var x   : byte;
    a, b: string;
begin
    write('Nhap xau thu nhat:');
    readln(a);
    write('Nhap xau thu hai:');
    readln(b);
    x:=length(b);
    {xác định do dài xau b để biết vị trí của kí tự cuối
    cùng}
    if a[1]=b[x] then write('Trung nhau')
        else write('Khac nhau');
    readln
end.
```

Khi chạy chương trình, nhập các xâu vào: nếu kí tự đầu tiên của xâu thứ nhất ‘thu’ không trùng với kí tự cuối cùng của xâu thứ hai ‘ha noi’ thì chương trình đưa ra thông báo: “Khac nhau”, ngược lại chương trình đưa ra thông báo: “Trung nhau”. Kết quả của chương trình cho như hình 48 dưới đây:



Hình 48. Kết quả chương trình kiểm tra 2 xâu

Ví dụ 3: Bài toán viết theo thứ tự ngược lại của xâu được nhập vào từ bàn phím.

Chương trình:

```
program vidu3;
var i, k: byte;
    a   : string;
begin
    write('Nhap xau:');
```

```

readln(a);
k:= length(a); {xac dinh do dai xau}
for i:= k downto 1 do write(a[i]);
readln
end.

```

Khi chạy chương trình, nhập vào xâu 'thu do ha noi' thì chương trình đưa ra kết quả "ion ah od uht", còn khi nhập vào xâu 'viet nam que huong toi' thì chương trình đưa ra kết quả: "iot gnouh euq man teiv" thì kết quả của chương trình cho như hình 49 dưới đây:



Hình 49. Kết quả chương trình viết theo thứ tự ngược lại của xâu được nhập vào

Ví dụ 4: Bài toán đưa ra màn hình xâu thu được bằng việc loại bỏ các dấu cách (nếu có) của xâu nhập vào từ bàn phím.

Chương trình:

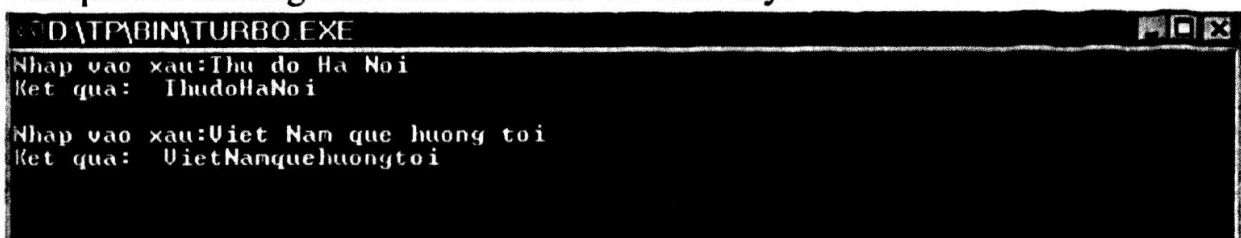
```

program vidu4;
var i, k:byte;
    a, b: string;
begin
    write('Nhap vao xau:');
    readln(a);
    k:= length(a)
    b:= ' '; {*Khoi tao xau rong*}
    for i:= 1 to k do
        if a[i]<> ' ' then b:= b + a[i];
        writeln('Ket qua: ', b);
    readln
End.

```

Khi chạy chương trình, nhập vào một xâu: 'Thu do Ha Noi' thì chương trình đưa ra kết quả: "ThudoHaNoi", còn khi nhập vào xâu 'Viet Nam que huong toi' thì chương trình đưa ra thông báo: "VietNamquehuongtoi".

Kết quả của chương trình cho như hình 50 dưới đây:



Hình 50. Kết quả chương trình đưa ra xâu không có dấu cách của một xâu

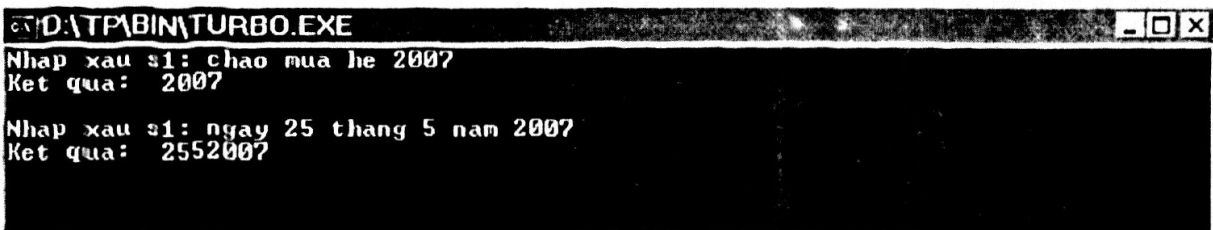
Ví dụ 5: Bài toán tạo chuỗi gồm tất cả các chữ số có trong chuỗi nhập vào từ bàn phím (giữ nguyên thứ tự xuất hiện của chúng) và đưa kết quả ra màn hình.

Chương trình:

```
program xulixau;  
var s1, s2: string;  
    i: byte;  
begin  
  write('Nhap xau s1: ');  
  readln(s1);  
  s2:= ''; {khởi tạo chuỗi s2 rỗng}  
  for i:= 1 to length(s1) do  
    if ('0'=<s1[i]) and (s1[i]<='9') then s2:=s2+s1[i];  
  writeln('Ket qua:',s2);  
  readln  
end.
```

Khi chạy chương trình, nhập vào chuỗi vừa ký tự vừa chữ số, chẳng hạn chuỗi: 'chao mua he 2007' thì kết quả của chương trình là 2007, còn khi nhập vào chuỗi 'ngay 25 thang 5 nam 2007', kết quả của chương trình là 2552007.

Kết quả của chương trình sau hai lần nhập chuỗi vào cho như hình 51 dưới đây:



Hình 51. Kết quả chương trình tạo chuỗi gồm tất cả các chữ số của chuỗi nhập vào



Bài tập và thực hành 5

1. Mục đích, yêu cầu

Làm quen với việc tìm kiếm, thay thế và biến đổi chuỗi.

2. Nội dung

Bài 1. Nhập vào từ bàn phím một chuỗi. Kiểm tra chuỗi đó có phải là chuỗi đối xứng hay không. Chuỗi đối xứng có tính chất đọc nó từ phải sang trái cũng như từ trái sang phải (còn được gọi là *chuỗi palindrome*).

a) Hãy chạy thử chương trình sau:

```

var i, x: byte;
    a, p: string;
begin
    write('Nhap vao xau: ');
    readln(a);
    x:= length(a); {xac dinh do dai xau}
    p:= ''; {khởi tạo xau rỗng}
    for i:= x downto 1 do
        p:= p+a[i]; {tao xau dao nguoc}
    if a= p then
        write('xau la palindrome')
    else
        write('xau khong la palindrome');
    readln
end.

```

b) Hãy viết lại chương trình trên, trong đó không cần có biến xâu *p*.

Bài 2. Viết chương trình nhập từ bàn phím một xâu kí tự *S* và thông báo ra màn hình số lần xuất hiện của mỗi chữ cái tiếng Anh trong *S* (không phân biệt chữ hoa hay chữ thường).

Bài 3. Nhập vào từ bàn phím một xâu. Thay thế tất cả các cụm kí tự 'amh' bằng cụm từ 'em'.

Hướng dẫn làm bài tập và thực hành 5

1. Mục đích, yêu cầu

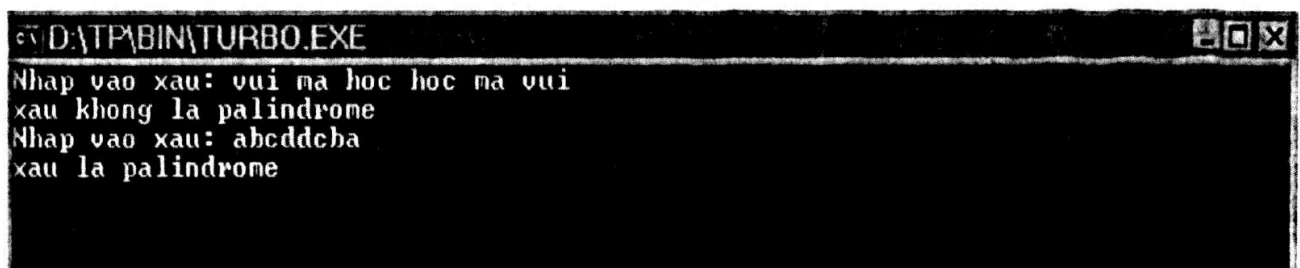
Làm quen với việc tìm kiếm, thay thế và biến đổi xâu.

2. Nội dung

Bài 1:

a) Khi chạy chương trình, nhập vào xâu: 'vui ma hoc hoc ma vui' thì chương trình đưa ra thông báo: "xau khong la palindrome", còn khi nhập vào xâu 'abceddcba' thì chương trình đưa ra thông báo: "xau la palindrome".

Kết quả của chương trình cho như hình 52 dưới đây:



Hình 52. Kết quả chương trình thông báo có phải là xâu palindrome?

b) Để viết lại chương trình mà dùng biến *p*, thì ta cần khai thác khả năng tham chiếu đến từng kí tự trong chuỗi thông qua vị trí của kí tự này. Như vậy, không cần thiết phải tạo một chuỗi mới để cuối cùng so sánh hai chuỗi, mà chỉ cần so sánh các cặp kí tự ở vị trí đối xứng nhau để kết luận chuỗi có là *palindrome* hay không.

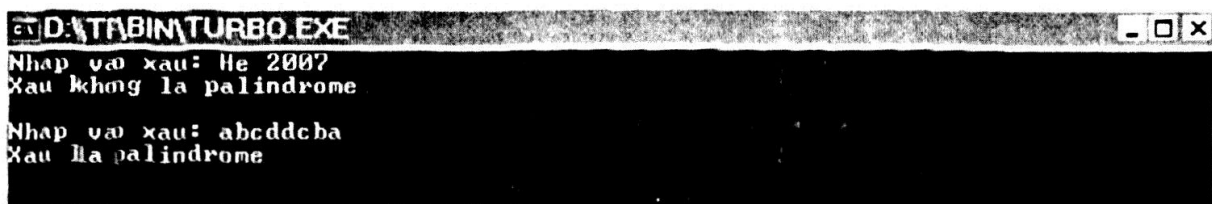
Bởi vậy, ta có thể dùng một biến logic để ghi nhận sự phát hiện này. Trước vòng lặp thực hiện các so sánh nói trên, cần khởi tạo cho biến logic đó, ở mỗi bước lặp, nếu hai kí tự được so sánh khác nhau thì biến logic đó sẽ phải thay đổi giá trị.

Chương trình sau đây dùng để kiểm tra xem chuỗi nhập vào có phải là chuỗi *palindrome* hay không:

```
var i, x: byte;
    a: string;
    palin: boolean;
begin
    write('Nhập vào chuỗi: ');
    readln(a);
    x := length(a); {xác định độ dài của chuỗi}
    palin := true;
    {khởi tạo palin, tạm coi chuỗi a là palindrome}
    for i := 1 to x div 2 do {so sánh cặp kí tự đối xứng}
        if a[i] <> a[x-i+1] then palin := false;
        if palin then writeln('Chuỗi là palindrome')
        else writeln('Chuỗi không là palindrome');
    readln
End.
```

Khi chạy chương trình, nhập vào chuỗi: 'He 2007' thì chương trình đưa ra thông báo: "chuỗi không là palindrome", còn khi nhập vào chuỗi 'abcdcba' thì chương trình đưa ra thông báo: "chuỗi là palindrome".

Kết quả của chương trình cho như hình 53 dưới đây:



Hình 53. Kết quả chương trình thông báo có phải là chuỗi *palindrome*?

❖ Tuy nhiên, ta có thể không dùng vòng *for-do* mà dùng *while-do* hay *repeat-until* và có thể không cần dùng biến logic. Chương trình sau đây đáp ứng yêu cầu đặt ra:

```
var i, x: byte;
    a: string;
    palin: boolean;
begin
```

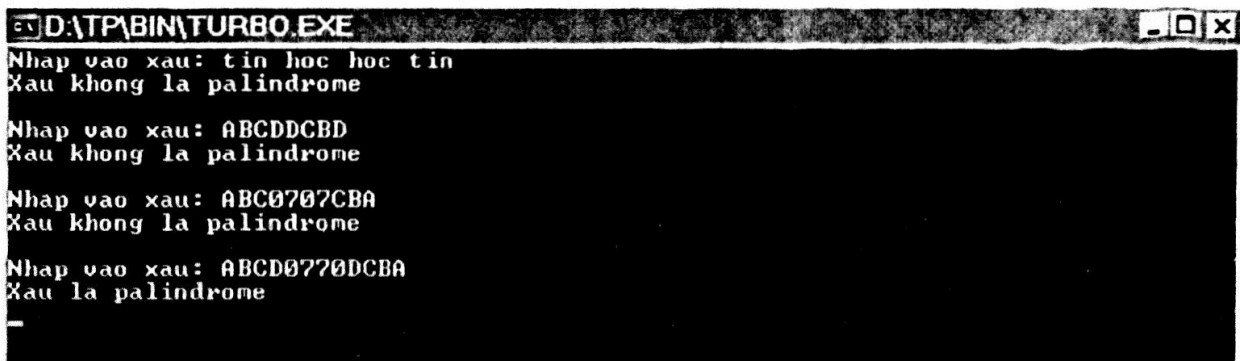
```

write('Nhap vao xau: ');
readln(a);
x:=length(a); {xac dinh do dai cua xa}
i:=1;
while (i<=(x div 2)) and (a[i]=a[x-i+1]) do i:=i+1;
  if i> (x div 2) then
    writeln('Xau la palindrome')
  else writeln('Xau khong la palindrome');
readln
End.

```

Khi chạy chương trình, nhập vào xâu: 'tin hoc hoc tin' thì chương trình đưa ra thông báo: "*xau khong la palindrome*", còn khi nhập vào xâu 'ABCDDCBD' thì chương trình cũng đưa ra thông báo: "*xau khong la palindrome*", còn khi nhập vào xâu: 'ABCD0770DCBA' thì chương trình đưa ra thông báo: "*xau la palindrome*".

Kết quả của chương trình cho như hình 55 dưới đây:



Hình 55. Kết quả chương trình thông báo có phải là xâu palindrome?

Bài 2. Để giải quyết bài toán này, chúng ta nhận thấy rằng:

- Cần ghi nhận số lần xuất hiện của từng chữ cái. Có tất cả 26 chữ cái 'A' 'Z'. Có thể dùng một mảng với chỉ số là kí tự từ 'A' đến 'Z' để ghi nhận số lần xuất hiện của các kí tự trong xâu S. Bởi vậy, chúng ta dùng một mảng một chiều để đếm số lần xuất hiện của một kí tự trong xâu S. Cụ thể, để ghi nhận số lần xuất hiện của kí tự, ta có thể dùng *dem['A']* để ghi nhận số lần xuất hiện kí tự A (hay kí tự *a*, vì không phân biệt chữ hoa hay chữ thường).
- Để giải quyết vấn đề không phân biệt chữ hoa hay chữ thường ta cần dùng hàm *Ucase(c)*.
- Do một kí tự xuất hiện trong xâu S có thể không phải là một chữ cái nên khi duyệt lần lượt từng kí tự trong xâu S, cần kiểm tra xem kí tự đó có phải là chữ cái hay không để ghi nhận số lần xuất hiện của nó. Chúng ta đã gặp đoạn chương trình kiểm tra một kí tự có là chữ số hay không ở ví dụ 5 tiết học 12. Từ đó, có thể viết được đoạn chương trình làm việc duyệt từng phần tử của xâu và đếm.
- Dàn ý của chương trình:


```

{phân khai báo}
begin
{nhập xâu S}
N:=length(S);
{Khởi tạo cho mảng Dem}
for i:=1 to N do
  {Nếu s[i] là chữ cái thì đếm tăng cho s[i]}
  for c:='A' to 'Z' do
    {Thông báo số lần xuất hiện của c}
  End.

```

• Chương trình nhập từ bàn phím một xâu kí tự S và thông báo ra màn hình số lần xuất hiện của mỗi chữ cái tiếng Anh trong S (không phân biệt chữ hoa hay chữ thường).

```

Program tinh_ki_tu;
var s,s1: string;
    i,j, n: integer;
    dem: array['A'..'Z'] of integer;
    c: char;
begin
  write('Nhap vao xau:'); readln(S);
  n:= length(s);
  for c:= 'A' to 'Z' do {khởi tạo cho mảng dem}
    dem[c]:= 0;
  s1:='';
  for i:=1 to n do {neu s[i] la chu cai thi dua vao xau s1}
    if (upcase(s[i])>='A') and (upcase(s[i])<='Z') then
      s1:=s1+upcase(s[i]);
  for c:='A' to 'Z' do
    for j:=1 to length(s1) do {dem so lan xuất hiện của từng
chu cai}
      if c= upcase(s1[j]) then dem[c]:= dem[c] + 1;
  for c:= 'A' to 'Z' do {in ra số lần xuất hiện của từng
chu cai}
    if dem[c]<>0 then
      writeln('so lan xuất hiện chu cai ',c,' la: ',dem[c]);
  readln
end.

```

Khi nhập vào lần lượt các xâu: 'dfd'2n5fv' 3m.A'; '55B7cfcManu07';
'8gs9'0A6ha5kQ' thì chương trình cho các kết quả như hình 56 dưới đây:

```

D:\TP\BIN\TURBO.EXE
Nhap vao xau:dfdf'2n5fv' 3m.A
so lan xuat hien chu cai A la: 1
so lan xuat hien chu cai D la: 2
so lan xuat hien chu cai F la: 2
so lan xuat hien chu cai M la: 1
so lan xuat hien chu cai N la: 1
so lan xuat hien chu cai V la: 1

Nhap vao xau:a5B7cfcManu07
so lan xuat hien chu cai A la: 2
so lan xuat hien chu cai B la: 1
so lan xuat hien chu cai C la: 2
so lan xuat hien chu cai F la: 1
so lan xuat hien chu cai M la: 1
so lan xuat hien chu cai N la: 1
so lan xuat hien chu cai U la: 1

Nhap vao xau:8gs90A6ha5kQ
so lan xuat hien chu cai A la: 2
so lan xuat hien chu cai G la: 1
so lan xuat hien chu cai H la: 1
so lan xuat hien chu cai K la: 1
so lan xuat hien chu cai Q la: 1
so lan xuat hien chu cai S la: 1

```

Hình 56. Kết quả chương trình tính số lần xuất hiện chữ cái của xâu

Bài 3. Đối với bài toán này:

- Để thay thế tất cả cụm từ "anh" trong một xâu *st* thành cụm kí tự "em", có thể làm một cách tự nhiên: Tìm vị trí xâu con "anh" trong xâu *st* đã cho, xóa xâu con này đi rồi chèn xâu "em" vào vị trí đó. Lặp đi lặp lại điều này cho đến khi không tìm thấy xâu "anh" cần thay thế trong xâu *st* nữa. Để giải quyết vấn đề này, chúng ta cần vận dụng các hàm *Pos*, thủ tục chuẩn *Delete*, *Insert*.

- Dàn ý chương trình:

{phần khai báo}

Begin

{Nhập xâu S}

{chừng nào còn tìm thấy xâu con 'anh' trong xâu S còn làm ba công việc sau:

- Tìm vị trí bắt đầu của xâu 'anh' ;
- Xóa xâu 'anh' vừa tìm thấy;
- Chèn xâu 'em' vào xâu S tại vị trí trước đây xuất hiện xâu "anh" ;

{in xâu S kết quả}

end.

- Chương trình nhập vào từ bàn phím một xâu, thay thế tất cả các cụm kí tự 'anh' bằng cụm từ 'em':

```

program thay_the_cum_tu;
var vt: byte;
    st: string;
Begin
    write('Nhap vao mot xau: ');
    readln(st);

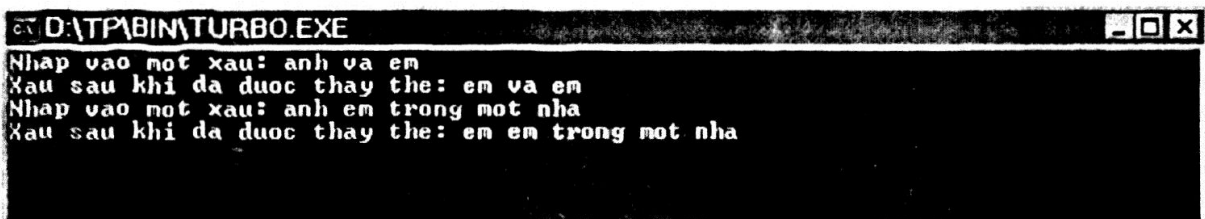
```

```

while pos('anh', st)<>0 do
begin
    vt:= pos('anh',st);
    delete(st,vt,3);
    insert('em',st,vt);
end;
write('Xau sau khi da duoc thay the: ',st);
readln
end.

```

Khi nhập vào lần lượt các xâu: 'anh va em'; 'anh em trong mot nha' thì chương trình cho các kết quả theo thứ tự như sau: 'em va em'; 'em em trong mot nha'. Kết quả chương trình cho như hình 57 dưới đây:



Hình 57. Kết quả chương trình thay thế 'anh' bằng 'em' của xâu

§13. KIỂU BÀN GHI

A. TÓM TẮT LÝ THUYẾT

- Dữ liệu kiểu *bàn ghi* dùng để mô tả các đối tượng có cùng một số thuộc tính mà các thuộc tính có thể có các kiểu dữ liệu khác nhau;
- *Kiểu bàn ghi* là một kiểu dữ liệu có cấu trúc. Một bàn ghi gồm các thành phần (gọi là *trường*), khác với các kiểu dữ liệu có cấu trúc khác (mảng và xâu), các trường có thể thuộc các kiểu dữ liệu khác nhau;
- Kiểu bàn ghi cho phép mô tả nhiều đối tượng có cùng một số thuộc tính, có hữu ích cho nhiều bài toán quản lý;
- Ngôn ngữ lập trình đưa ra quy tắc, cách thức xác định:
 - Tên kiểu bàn ghi;
 - Tên các thuộc tính (trường);
 - Kiểu dữ liệu của mỗi trường;
 - Cách khai báo biến;
 - Cách tham chiếu đến trường.

1. Khai báo

- ◊ Các thông tin cần khai báo bao gồm tên kiểu bàn ghi, tên các thuộc tính, kiểu dữ liệu của mỗi thuộc tính.

◊ Kiểu bản ghi thường được định nghĩa như sau:

```
type <tên kiểu bản ghi> = record
    <tên trường 1>: <kiểu trường 1>
    .....
    <tên trường m>: <kiểu trường m>
end;
```

Sau khi có kiểu bản ghi, biến kiểu bản ghi có thể được khai báo như sau:

```
var
    <tên biến bản ghi>: <tên kiểu bản ghi>;
```

Ví dụ

Trong chương trình xử lý kết quả tốt nghiệp có thể sử dụng khai báo sau đây:

```
const Max= 60; {gia thiet si so lop cao nhat la 60}
type
    Hocsinh = record
        HoTen: string[30];
        NgaySinh: string[10];
        GioiTinh: boolean;
        Tin, Toan, Li, Hoa, Van, Su, Dia: Real;
    End;
```

```
var
    A, B: HocSinh;
    Lop: array[1..Max] of HocSinh;
```

- Nếu *A* là biến kiểu bản ghi và *X* là tên một trường của *A*, thì để tham chiếu đến trường *X*, ta viết:

A.X

- Để tham chiếu đến điểm thi tin học của học sinh *A* ta viết:

A.Tin

2. Gán giá trị

- *Có hai cách để gán giá trị cho biến bản ghi:*

Cách 1: Dùng lệnh gán trực tiếp: Nếu *A* và *B* là hai biến bản ghi cùng kiểu, thì ta có thể gán giá trị của *B* cho *A* bằng câu lệnh:

A := B;

Cách 2: Gán giá trị cho từng trường: có thể thực hiện bằng lệnh gán hoặc nhập từ bàn phím.

Ví dụ: Chương trình nhập vào từ bàn phím thông tin của từng học sinh trong lớp, thực hiện xếp loại và đưa ra màn hình kết quả xếp loại học sinh:

```
program xep_loai;
```

```

uses crt;
const max= 60;
type Hocsinh = record
    hoten: string[30];
    ngaysinh: string[10];
    Diachi: string[50];
    Toan, Van: real;
    Xeploai: char;
end;

var
    Lop: array[1..max] of hocsinh;
    N,i: byte;
Begin
    clrscr;
    write('So luong hoc sinh trong lop N='); readln(N);
    for i:= 1 to N do
        begin
            writeln('Nhap so lieu ve hoc sinh thu',i,': ');
            write('Ho va ten: '); readln(Lop[i].hoten);
            write('Ngay sinh: '); readln(Lop[i].ngaysinh);
            write('Dia chi : '); readln(Lop[i].Diachi);
            write('Diem Toan: '); readln(Lop[i].Toan);
            write('Diem Van : '); readln(Lop[i].Van);
            if Lop[i].Toan+Lop[i].Van>=18
                then Lop[i].xeploai:='A';
            if (Lop[i].Toan+Lop[i].Van>=14) and
(Lop[i].Toan+Lop[i].Van<18)
                then Lop[i].Xeploai:='B';
            if (Lop[i].Toan+Lop[i].Van>=10) and
(Lop[i].Toan+Lop[i].Van<14)
                then Lop[i].Xeploai:='C';
            if (Lop[i].Toan+Lop[i].Van<=10)
                then Lop[i].xeploai:='D';
        end;
    clrscr;
    writeln('Danh sach xep loai hoc sinh trong lop: ');
    for i:= 1 to N do
        writeln(Lop[i].Hoten:30,'- Xep loai:', Lop[i].Xeploai);
    readln
End.

```

Kết quả của chương trình sau khi nhập dữ liệu của 10 em học sinh như hình 58 dưới đây:

```
D:\TP\BIN\TURBO.EXE
Danh sach xep loai hoc sinh trong lop:
      Tran Anh - Xep loai: B
      Tran Bay - Xep loai: C
      Nguyen Dung - Xep loai: B
      Cao Cuong - Xep loai: B
      Hoang Ha - Xep loai: B
      Hoang Ky - Xep loai: C
      Tran Thao - Xep loai: A
      Pham Tuan - Xep loai: C
      Hoang Tung - Xep loai: C
      Nguyen Thi Hoang Yen - Xep loai: B
```

Hình 58. Kết quả của chương trình xếp loại học sinh

TÓM TẮT CHƯƠNG IV

- **Kiểu dữ liệu có cấu trúc** được xây dựng từ những kiểu dữ liệu đã có theo quy tắc, khuôn dạng do ngôn ngữ lập trình cung cấp;
- **Mảng một chiều**
 - Mảng một chiều là dãy hữu hạn các phần tử cùng kiểu;
 - Khai báo: tên mảng, kiểu chỉ số, kiểu phần tử;
 - Tham chiếu phần tử mảng: *tên biến mảng[chi số phần tử]*
- **Mảng hai chiều**
 - Mảng hai chiều là bảng các phần tử cùng kiểu;
 - Khai báo: tên mảng, kiểu chỉ số dòng, kiểu chỉ số cột, kiểu phần tử;
 - Tham chiếu phần tử mảng: *tên biến mảng[chi số dòng, chi số cột]*
- **Kiểu dữ liệu xâu**
 - Xâu là dãy các kí tự trong bảng mã ASCII.
 - Các thao tác xử lí thường xử dụng:
 - Phép ghép xâu;
 - Phép so sánh;
 - Các thủ tục và hàm chuẩn.
- **Kiểu bản ghi**
 - Khai báo: tên bản ghi, tên và kiểu các trường;
 - Tham chiếu trường của bản ghi: *tên biến bản ghi.tên trường*

B. CÂU HỎI VÀ BÀI TẬP

I. BÀI TẬP CƠ BẢN

1. Tại sao mảng là kiểu dữ liệu có cấu trúc?
2. Tại sao phải khai báo kích thước của mảng?
3. Các phần tử của mảng có thể có những kiểu gì?

4. Tham chiếu đến phần tử của mảng bằng cách nào?
5. Viết chương trình nhập vào từ bàn phím số nguyên dương $N(N \leq 1000)$ và dãy A gồm N số nguyên $A_1, A_2, \dots, A_N$ có giá trị tuyệt đối không lớn hơn 1000. Hãy cho biết dãy A có phải là một cấp số cộng hay không và thông báo kết quả ra màn hình.
6. Viết chương trình nhập vào từ bàn phím số nguyên dương $N(N \leq 100)$ và dãy A gồm N số nguyên $A_1, A_2, \dots, A_N$ có giá trị tuyệt đối không lớn hơn 100. Hãy đưa ra những thông tin sau:
Số lượng số chẵn và số lẻ trong dãy;
Số lượng số nguyên tố trong dãy;
7. Dãy F là dãy *Phi-bô-na-xi* nếu:
 $F_1=0; F_2=1; F_N = F_{N-1} + F_{N-2}$ với $N \geq 2$.
Viết chương trình nhập vào từ bàn phím số nguyên dương N và đưa ra màn hình số hạng N của dãy *Phi-bô-na-xi*. Chương trình của em thực hiện được với giá trị lớn nhất của N là bao nhiêu?

8. Chương trình sau đây thực hiện những gì?

```

Program BT8;
const NMax = 50;
type Mass = array[1..NMax, 0..NMax-1] of real;
var A: Mass;
i, j, N: byte; C: real;
Begin
  Write('Nhap N= ?'); readln(N);
  for i:= 1 to N do
    for j:= 0 to N-1 do
      begin
        write('A[', i, ', ', j, '], =');
        readln(A[i, j])
      end;
  for i:= 1 to N do
    for j:= 1 to N-1 do
      begin
        C:= A[i, j];
        A[i, j]:= A[N-i+1, j];
        A[N-i+1, j]:= C;
      end;
  for i:=1 to N do
    begin
      for j:=1 to N-1 do write(A[i, j]:5:2, ' ');
      writeln
    end;
End.

```


9. Cho mảng hai chiều kích thước $n \times m$ với các phần tử là những số nguyên. Tìm trong mỗi dòng phần tử lớn nhất rồi đổi chỗ nó với phần tử có chỉ số dòng bằng chỉ số cột.

Chương trình sau đây giải bài toán trên:

```
program Diag;
var
  N, i, j, max, Ind, Vsp: integer;
  A: array[1..15, 1..15] of integer;
begin
  write('Nhap N:'); readln(N);
  for i:= 1 to N do
    for j:= 1 to N do
      begin
        write('A[' , i, ', ' , j, ']= ');
        readln(A[i,j]);
      end;
    for i:=1 to N do
      begin
        Max:= A[i,1]; Ind:= 1;
        for j:= 2 to N do
          if A[i,j] > Max then
            begin
              Max:= A[i,j]; Ind:= j;
            end;
        Vsp:=A[i,i]; A[i,i]:=Max; A[i,Ind]:=Vsp;
      end;
    for i:= 1 to N do
      begin
        writeln;
        for j:= 1 to N do write(A[i,j]:3);
      end;
    writeln
  End.
```

Hãy sửa lại chương trình trên khi yêu cầu bài toán thay dòng bằng cột.

10. Viết chương trình nhập vào từ bàn phím chuỗi ký tự S có độ dài không quá 100. Hãy cho biết có bao nhiêu chữ số thập phân xuất hiện trong chuỗi S . Thông báo kết quả ra màn hình.
11. Hãy bổ sung thêm chương trình *xep_loai* (ở §13) những lệnh cần thiết để chương trình đưa ra danh sách học sinh xếp loại A .

Hướng dẫn trả lời câu hỏi và bài tập

1. Mảng là kiểu dữ liệu có cấu trúc bởi vì mảng (một chiều, hai chiều hay nhiều chiều) là kiểu có cấu trúc được đề cập tới sớm nhất trong các ngôn ngữ lập trình. Nó được xây dựng từ những kiểu dữ liệu đã có theo quy tắc khuôn dạng do ngôn ngữ lập trình cung cấp. Nó được dùng để chỉ định một nhóm đối tượng cùng một tính chất nào đó. Chẳng hạn, vectơ là một nhóm các số mà mỗi số ta có thể xác định chỉ cần biết chỉ số. Như vậy, để khai báo kiểu mảng phải chỉ ra kiểu dữ liệu chung của các phần tử và kiểu chỉ số.

Chúng ta phải khai báo kích thước của mảng bởi vì để cách đánh số các phần tử của nó.

2. Các phần tử của mảng có thể có những kiểu sau đây: *real*, *boolean*, *integer*, *longint*

3. Tham chiếu đến phần tử của *mảng một chiều* được xác định bởi tên mảng cùng với chỉ số, được viết trong cặp ngoặc [và] (ví dụ *nhietdo[20]*). Còn tham chiếu đến phần tử của *mảng hai chiều* được xác định bởi tên mảng cùng với hai chỉ số được phân cách bởi dấu phẩy và viết trong cặp ngoặc [và] (ví dụ *ArrInt[5, 9]*).

4. Để lập chương trình nhập từ bàn phím số nguyên dương N ($N \leq 100$) và dãy A gồm N số nguyên $A_1, A_2, \dots, A_N$, có giá trị tuyệt đối không lớn hơn 100, thì ta cần phải thực hiện việc kiểm tra xem số N nhập vào (số phần tử của dãy) có thỏa mãn yêu cầu của đầu bài hay không nghĩa là $0 \leq N \leq 100$, nếu không đưa ra thông báo “*Mời nhập lại*”. Còn các phần tử $A_1, A_2, \dots, A_N$ của mảng cũng phải là những số nguyên và có giá trị tuyệt đối không lớn hơn 100 ($A[i] \geq -100$ và $A[i] \leq 100$), nếu không đưa ra thông báo “*Mời nhập lại*”

Tiếp đến ta phải kiểm tra dãy A nhập vào có phải là một cấp số cộng hay không nghĩa là các phần tử của dãy A vừa nhập vào có làm thành một cấp số cộng hay không và đưa ra màn hình thông báo kết quả ra màn hình “*A là cấp số cộng*” hoặc “*A không là cấp số cộng*”. Bởi vậy,

Lấy $d = A[2] - A[1]$

Khi đó, dãy A là cấp số cộng nếu thỏa mãn điều kiện:

$$A[i] = A[1] + (i-1)d.$$

(hoặc điều kiện $A[i] - A[i-1] = d$ với $1 < i \leq N$).

Trong chương trình ta dùng một vòng lặp theo biến đếm i để kiểm tra xem mỗi $A[i]$ có thỏa mãn điều kiện nói trên hay không, chỉ cần phát hiện được một phần tử của A không thỏa mãn là kết luận được dãy A không là cấp số cộng.

Chương trình:

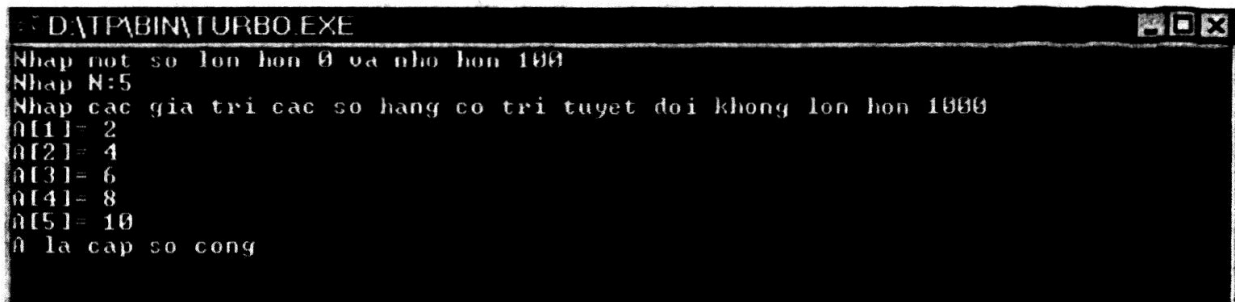
```
program bai5_chuong4;
uses crt;
var
  A: array[1..100] of integer;
  d, n, i, dem: integer;
```

```

begin
  clrscr;
  writeln('Nhap mot so lon hon 0 va nho hon 100');
  repeat
    write('Nhap N:'); readln(N);
  if (N>=1000) or (N<=0 ) then writeln('Moi nhap lai !');
  until (N>0) and (N<1000);
  writeln('Nhap cac gia tri cac so hang co tri tuyet doi
  khong lon hon 1000');
  for i:=1 to n do
    repeat
      write('a[' ,i, ']='); readln(a[i]);
    if (A[i]>100) or (A[i]<-1000) then
      write('Moi nhap lai ');
    until (A[i]>-1000) and (A[i]<1000);
    {Kiem tra xem co la cap so cong khong?}
    d:= A[2] - A[1];
    dem:=0;
    for i:= 2 to n do
      if A[i]- A[i-1]<>d then dem:= dem + 1;
    if dem >0 then writeln('A khong la cap so cong')
      else writeln('A la cap so cong');
    readln;
  end.

```

Khi chạy chương trình, nếu nhập số phần tử của dãy $N=5$ và các phần tử của dãy theo thứ tự là 2, 4, 6, 8, 10 thì chương trình đưa ra thông báo “A la cap so cong”(Hình 59).



```

D:\TP\BIN\TURBO EXE
Nhap mot so lon hon 0 va nho hon 100
Nhap N:5
Nhap cac gia tri cac so hang co tri tuyet doi khong lon hon 1000
a[1]= 2
a[2]= 4
a[3]= 6
a[4]= 8
a[5]= 10
A la cap so cong

```

Hình 59. Kết quả chương trình với $N=5$

Nhưng khi nhập N không thỏa mãn điều kiện đầu bài, chẳng hạn $N= -5$ thì chương trình đưa ra thông báo “Moi nhap lai!”. Khi đó, ta phải vào lại N , chẳng hạn, $N=5$ và nhập tiếp các phần tử của dãy, chẳng hạn: 1, 3, 5, 7, 9 thì kết quả của chương trình: “A la cap so cong” (Hình 60).

```
D:\T\BIN\TURBO.EXE
Nhap mot so lon hon 0 va nho hon 100
Nhap N:5
Moi nhap lai ?
Nhap N:5
Nhap cac gia tri cac so hang co tri tuyet doi khong lon hon 1000
A[1]= ?
A[2]= ?
A[3]= ?
A[4]= ?
A[5]= ?
A la cap so cong
```

Hình 61. Kết quả chương trình với $N=5$

Còn trong trường hợp, các phần tử nhập vào của dãy không thỏa mãn điều kiện $A[i] < 1000$ và $A[i] > -1000$: thì ta phải nhập lại các phần tử của dãy cho đến khi chương trình đưa ra thông báo A là cấp số cộng hay không?

Chẳng hạn, ta nhập vào dãy $N=7$ phần tử với giá trị các phần tử lần lượt như sau: 2, 5, 7, 8, 9, 4, 7 (ở phần tử thứ tư lúc đầu nhập vào số -1500 thì chương trình yêu cầu nhập lại và đã nhập vào số 8) thì chương trình đưa ra thông báo: " A không là cấp số cộng" (Hình 62).

```
D:\T\BIN\TURBO.EXE
Nhap mot so lon hon 0 va nho hon 100
Nhap N:7
Nhap cac gia tri cac so hang co tri tuyet doi khong lon hon 1000
A[1]= ?
A[2]= ?
A[3]= ?
A[4]= -1500
Moi nhap lai A[4]= 8
A[5]= ?
A[6]= ?
A[7]= ?
A khong la cap so cong
```

Hình 62. A không là cấp số cộng

5. Đối với bài toán này thì phần kiểm tra số nguyên dương $N(N \leq 100)$ và dãy A gồm N số nguyên $A_1, A_2, \dots, A_N$, có giá trị tuyệt đối không lớn hơn 1000 giống như bài tập 5 đã nêu ở trên. Chúng ta chỉ cần giải quyết thêm 2 vấn đề nữa, đó là đếm số lượng số chẵn, số lẻ và số lượng số nguyên tố trong dãy.

Chương trình:

```
program baitap6_chuong4;
uses crt;
var  A: array[1..100] of integer;
     NT: boolean;
     N, i, j: integer;
     so_nt, so_chan: integer;
begin
  for i:= -1000 to 1000 do
    if i>0 then NT:= false;
```

```

so_chan:=0; so_nt:=0;
{Nhap vao}
repeat
    write('So phan tu cua day A (N<=100), N= ');
    readln(N);
until (N>0) and (N<=100);
for i:= 1 to N do
begin
    {kiem tra cac phan tu cua day khi nhap vao}
    repeat
        write('A[' ,i,']= ');readln(A[i]);
        if (a[i]>1000) or (a[i]<-1000) then
            write('Moi nhap lai ');
    until (a[i]>-1000) and (a[i]<1000); =
    if A[i] mod 2 =0 then so_chan:= so_chan + 1;
    if A[i] >1 then
        begin
            u:= 2;
            while ((u<=sqrt(A[i])) and (A[i] mod u<>0)) do
                u:= u + 1;
            if u>sqrt(A[i]) then
                so_nt:= so_nt + 1;
        end;
    end;
end;
{In ra man hinh}
writeln('So luong so chan:',so_chan);
writeln('So luong so le:',N - so_chan);
writeln('So luong so nguyen to:',so_nt);
readln
End.

```

Khi chạy chương trình, ta lần lượt nhập các phần tử của dãy A :

❖ Với $N = 3$ và các phần tử được nhập vào theo thứ tự:

$A[1]=5$

$A[2]=7$

$A[3]=900$

thì chương trình đưa ra thông báo:

So luong so chan: 1

So luong so le: 2

So luong so nguyen to: 2

❖ Với $N= 5$ và các phần tử được nhập vào theo thứ tự:

$A[1]=12$

$A[2]=3$

$A[3]=5$

A[4]=10

A[5]=11

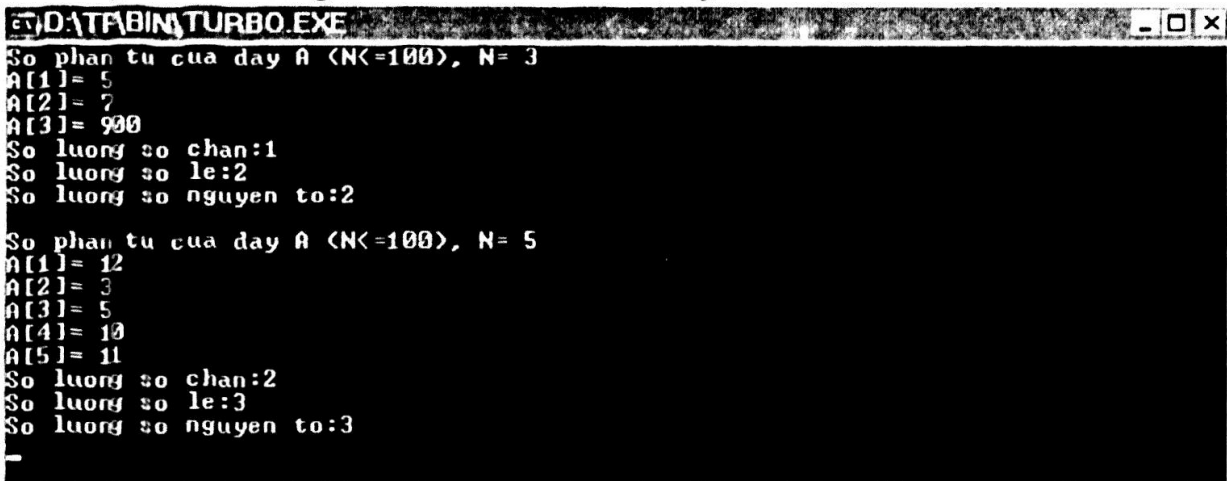
thì chương trình đưa ra thông báo:

Số lượng số chẵn: 2

Số lượng số lẻ: 3

Số lượng số nguyên tố: 3

Kết quả của chương trình như hình 63 sau đây:



```
DATA\BIN\TURBO.EXE
Số phần tử của dãy A (N<=100), N= 3
A[1]= 5
A[2]= 7
A[3]= 999
Số lượng số chẵn:1
Số lượng số lẻ:2
Số lượng số nguyên tố:2

Số phần tử của dãy A (N<=100), N= 5
A[1]= 12
A[2]= 3
A[3]= 5
A[4]= 10
A[5]= 11
Số lượng số chẵn:2
Số lượng số lẻ:3
Số lượng số nguyên tố:3
```

Hình 63. Kết quả của chương trình đếm số lượng số chẵn, số lẻ và số nguyên tố

❖ Trong trường hợp số phần tử nhập vào và giá trị các phần tử của dãy A không thỏa mãn điều kiện thì chương trình sẽ có thông báo mời nhập lại. Chẳng hạn, trong các trường hợp sau đây:

A[1]= 12

A[2]= 2000 thì chương trình đưa ra yêu cầu

Mời nhập lại A[2]= 800

A[3]= 17

A[4]= -1500 thì chương trình đưa ra yêu cầu

Mời nhập lại A[4]= -200

A[5]=31

A[6]= 5

A[7]= 350

Kết quả chương trình đưa ra thông báo (Hình 64):

Số lượng số chẵn: 4

Số lượng số lẻ: 3

Số lượng số nguyên tố: 3

```
D:\TP\BIN\TURBO.EXE
Số phần tử của dãy A (N<=100), N= -3
Nhập lại, số phần tử của dãy A (N<=100), N=7
A[1]= 12
A[2]= 2000
Moi nhap lai A[2]= 800
A[3]= 17
A[4]= -1500
Moi nhap lai A[4]= -200
A[5]= 31
A[6]= 5
A[7]= 350
Số lượng số chẵn:4
Số lượng số lẻ:3
Số lượng số nguyên tố:3
```

Hình 64

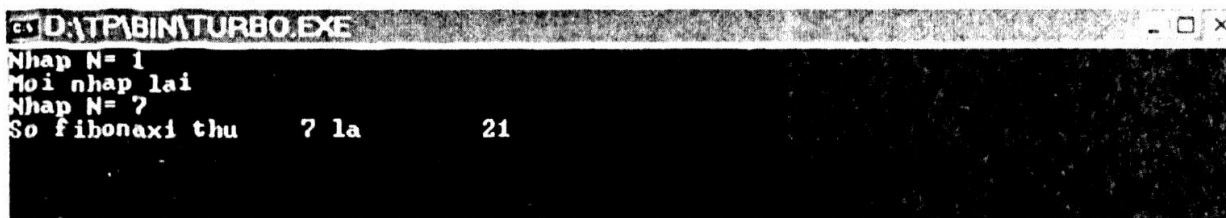
7. Ta cần viết chương trình nhập vào từ bàn phím số nguyên dương N và đưa ra màn hình số hạng thứ N của dãy *Phi-bô-nan-xi* và đưa ra kết luận: với giá trị lớn nhất nào của N thì chương trình thực hiện được?

- *Chương trình:*

```
program bai7_chuong4;
uses crt;
var
  N,i: word;
  F,F1,F2: word;
Begin
clrscr;
repeat
  writeln('Tim so hang thu N cua day Fibonaxi, N=');
  readln(N);
  if N<=2 then writeln('Moi nhap lai !');
until N>2;
  F1:= 1;
  F2:= 2;
  for i:= 3 to N do
    begin
      F:= F1 + F2;
      F1:= F2;
      F2:= F;
    end;
  writeln('So fibonaxi thu', N:5,'la',F:10);
  readln;
End.
```


Khi chạy chương trình, nếu nhập $N < 2$, chẳng hạn $N = 1$ thì trên chương trình đưa ra thông báo "Moi nhap lai". Còn khi nhập $N \geq 2$, chẳng hạn $N = 7$ thì chương trình đưa ra kết quả "So fibonaxi thu 7 la 21".

Kết quả chương trình đưa ra kết quả như hình 65 dưới đây:



Hình 65. Kết quả chương trình với $N = 7$.

- Chương trình trên chỉ chạy được với $N = 1001$ vì số Fi-bô-na-xi thứ 1001 là 65048, số số Fi-bô-na-xi thứ 1002 vượt quá phạm vi của kiểu *word*.

8. Chương trình sau đây thực hiện những gì?

```
program BT8;
const Nmax= 50;
type Mass = array[1..Nmax, 0..Nmax-1] of real;
var A: Mass;
    i, j, N: byte; C: real;
begin
    write('Nhap N=?'); readln(N);
    for i:= 1 to N do
        for j:= 1 to N-1 do
            begin
                write('A[' , i , ', ' , j , ']='); readln(A[i,j]);
            end;
    for i:= 1 to N do
        for j:= 1 to N-1 do
            begin
                C:= A[i,j];
                A[i,j]:= A[N-i+1,j];
                A[N-i+1,j]:= C;
            end;
    for i:= 1 to N do
        begin
            for j:= 1 to N-1 do write(A[i,j]:5:2, ' ');
            writeln
        end;
end.
```

Khi chạy chương trình, với $N = 4$ thì kết quả của chương trình như hình 67 dưới đây:

```

D:\TP\BIN\TURBO.EXE
Nhap N=?4
A[1,1]=9
A[1,2]=7
A[1,3]=8
A[2,1]=2
A[2,2]=6
A[2,3]=3
A[3,1]=8
A[3,2]=6
A[3,3]=9
A[4,1]=5
A[4,2]=8
A[4,3]=4
9.00 7.00 8.00
2.00 6.00 3.00
8.00 6.00 9.00
5.00 8.00 4.00

```

Hình 67. Kết quả của chương trình với $N = 4$

- Chương trình thực hiện việc hoán đổi vị trí dòng thứ i với dòng thứ $N-i+1$, nghĩa là hoán đổi vị trí dòng đầu tiên với dòng cuối cùng của mảng hai chiều, dòng thứ hai từ trên xuống với dòng thứ hai từ dưới lên,.. Việc hoán đổi vị trí dòng thứ i với dòng đối xứng với nó được thực hiện khi i nhận giá trị từ 1 đến N , làm cho mỗi dòng được hoán đổi vị trí hai lần. Vì vậy, cuối cùng mảng A không thay đổi so với ban đầu.

9. Chương trình tìm trong mỗi dòng phần tử lớn nhất rồi đổi chỗ nó với phần tử có chỉ số dòng bằng số cột tức là chương trình

```

program Diag;
var
  N, i, j, max, Ind, Vsp: integer;
  A: array[1..15, 1..15] of integer;
begin
  write('Nhap N:'); readln(N);
  for i:= 1 to N do
    for j:= 1 to N do
      begin
        write('A[' , i , ', ' , j , ']= ');
        readln(A[i,j]);
      end;
  for i:= 1 to N do
    begin
      Max:= A[i,1]; Ind:= 1;
      for j:= 2 to N do
        if A[i,j] > Max then
          begin
            Max:= A[i,j]; Ind:= j;
          end;
      Vsp:= A[i,i]; A[i,i]:= Max; A[i,Ind]:= Vsp;
    end;
  for i:= 1 to N do

```

```

begin
    writeln;
    for j:= 1 to N do write(A[i,j]:3);
end;
writeln
End.

```

có thể được chia thành *ba đoạn chương trình* sau đây:

- **Đoạn thứ nhất:** Hai vòng lặp *for-do* lồng nhau ở đầu chương trình có nhiệm vụ nhập vào một mảng hai chiều từ bàn phím

```

for i:= 1 to N do
    for j:= 1 to N do
        begin
            write('A[' , i , ' , ' , j , ']= ');
            readln(A[i,j]);
        end;

```

- **Đoạn thứ hai:** Hai vòng lặp lồng nhau tiếp theo thực hiện việc tìm phần tử lớn nhất trên dòng thứ *i* hoán đổi vị trí phần tử này với phần tử vừa nằm trên dòng *i* vừa có chỉ số cột bằng *i*.

```

for i:= 1 to N do
    begin
        Max:= A[i,1]; Ind:= 1;
        for j:= 2 to N do
            if A[i,j] > Max then
                begin
                    Max:= A[i,j]; Ind:= j;
                end;
        Vsp:= A[i,i]; A[i,i]:= Max; A[i,Ind]:= Vsp;
    end;

```

- **Đoạn thứ ba:** Hai vòng lặp lồng nhau cuối chương trình in ra mảng kết quả

```

for i:=1 to N do
    begin
        writeln;
        for j:= 1 to N do write(A[i,j]:3);
    end;
writeln

```

***)** Khi chạy chương trình trên, nhập vào $N=3$ thì ta có mảng hai chiều 3×3 với 9 phần tử, chẳng hạn theo thứ tự như sau:

```

A[1,1]= 4
A[1,2]= 6
A[1,3]= 8
A[2,1]= 3

```

$$A[2,2]=7$$

$$A[2,3]=9$$

$$A[3,1]=4$$

$$A[3,2]=9$$

$$A[3,3]=5$$

Ta nhận thấy rằng, ở dòng thứ nhất, phần tử lớn nhất của dòng là $A[1,3]=8$, phần tử có chỉ số dòng bằng chỉ số cột là $A[1,1]=4$. Bởi vậy, sau khi trao đổi thì giá trị của $A[1,1]=8$, còn $A[1,3]=4$. Vì vậy, dòng thứ nhất sau khi được trao đổi là: $A[1,1]=8$, $A[1,2]=6$, $A[1,3]=4$.

Tương tự, ở dòng thứ 2 thì phần lớn nhất của dòng là $A[2,3]=9$ sẽ được trao đổi với phần tử $A[2,2]=7$. Bởi vậy, sau khi trao đổi thì giá trị của $A[2,2]=9$, còn $A[2,3]=7$. Vì vậy, dòng thứ hai sau khi được trao đổi là:

$$A[2,1]=3, A[2,2]=9, A[2,3]=7.$$

Ở dòng thứ ba sau khi được trao đổi là: $A[3,1]=4$, $A[3,2]=5$, $A[3,3]=9$.

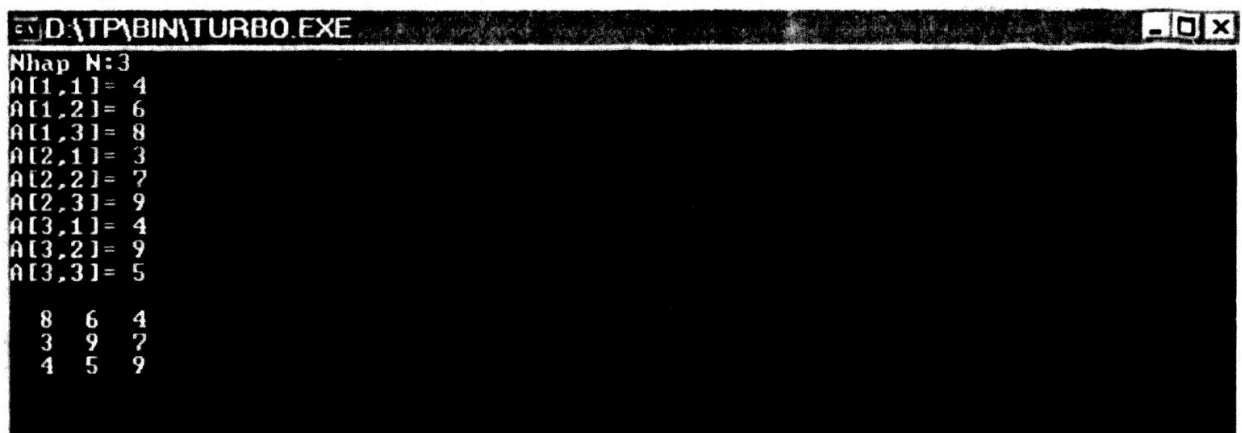
Khi đó, các dòng sau khi được trao đổi sẽ như sau:

$$A[1,1]=8, A[1,2]=6, A[1,3]=4$$

$$A[2,1]=3, A[2,2]=9, A[2,3]=7$$

$$A[3,1]=4, A[3,2]=5, A[3,3]=9$$

Kết quả chương trình có dạng như hình 68 dưới đây:



```

D:\TP\BIN\TURBO.EXE
Nhap N:3
A[1,1]= 4
A[1,2]= 6
A[1,3]= 8
A[2,1]= 3
A[2,2]= 7
A[2,3]= 9
A[3,1]= 4
A[3,2]= 9
A[3,3]= 5

  8  6  4
  3  9  7
  4  5  9
  
```

Hình 68. Mảng hai chiều 3x3 sau khi được trao đổi

*) Muốn sửa lại chương trình để thực hiện đổi chỗ phần tử lớn nhất trên cột với phần tử vừa nằm trên cột đó vừa có chỉ số dòng bằng chỉ số cột thì chỉ cần hoán đổi vị trí của hai chỉ số trong đoạn thứ hai của chương trình. Điều đó có nghĩa là chỉ đổi i trở thành chỉ số thứ hai, chỉ số thứ hai trở thành chỉ số i trong các câu lệnh có liên quan. Như vậy, chỉ có đoạn thứ hai trong chương trình khác đi và trở thành như sau:

```

for i:= 1 to N do
begin
    Max:= A[1,i];
    Ind:= 1;
    for j:= 2 to N do
  
```

```

        if A[i,j] > Max then
            begin
                Max:= A[j,i];
                Ind:= j;
            end;
        Vsp:= A[i,i]; A[i,i]:= Max; A[Ind,i]:= Vsp;
{Hoac Vsp:=A[i,i];A[i,i]:=A[Ind,i]; A[Ind,i]:=Vsp;}
    end;

```

Do vậy, chương trình tìm trong mỗi cột phần tử lớn nhất rồi đổi chỗ nó với phần tử có chỉ số dòng bằng số cột là như sau:

```

program Diag_2;
var
    N, i, j, Max, Ind, Vsp: integer;
    A: array[1..15, 1..15] of integer;
begin
    write('Nhap N:'); readln(N);
    for i:= 1 to N do
        for j:= 1 to N do
            begin
                write('A[' ,i ,',',',j ,']= ');
                readln(A[i,j]);
            end;
        for i:= 1 to N do
            begin
                Max:= A[1,i]; Ind:= 1;
                for j:= 2 to N do
                    if A[j,i] > Max then
                        begin
                            Max:= A[j,i];
                            Ind:= j;
                        end;
                Vsp:= A[i,i]; A[i,i]:= Max; A[Ind,i]:= Vsp;
{Hoac Vsp:=A[i,i];A[i,i]:=A[Ind,i]; A[Ind,i]:=Vsp;}

            end;
        for i:= 1 to N do
            begin
                writeln;
                for j:= 1 to N do write(A[i,j]:3);
            end;
        writeln
        readln
    end
end

```

Khi chạy chương trình, giả sử nhập $N=3$ với các phần tử của mảng theo thứ tự như sau:

$A[1,1]=4$ $A[1,2]=3$ $A[1,3]=6$
 $A[2,1]=9$ $A[2,2]=5$ $A[2,3]=7$
 $A[3,1]=5$ $A[3,2]=9$ $A[3,3]=4$

Ta nhận thấy rằng, phần tử lớn nhất của cột thứ nhất là 9 và nó sẽ trao đổi với phần tử $A[1,1]=4$. Tương tự, phần tử lớn nhất của cột thứ hai là 9 và nó sẽ trao đổi với phần tử $A[2,2]=5$. Phần tử lớn nhất của cột thứ ba là 7 và nó sẽ trao đổi với phần tử $A[3,3]=4$.

Như vậy, các phần tử của mảng A sau khi đã trao đổi như sau:

9	3	6
4	9	4
5	5	7

Kết quả chương trình như hình 69 dưới đây:

```

D:\TP\BIN\TURBO.EXE
Nhap N:3
A[1,1]= 4
A[1,2]= 3
A[1,3]= 6
A[2,1]= 9
A[2,2]= 5
A[2,3]= 7
A[3,1]= 5
A[3,2]= 9
A[3,3]= 4

9 3 6
4 9 4
5 5 7
  
```

Hình 69. Kết quả chương trình với $N=3$ sau khi đã trao đổi

10. Chương trình nhập vào từ bàn phím chuỗi ký tự S có độ dài không quá 100. Cho biết số chữ số thập phân xuất hiện trong chuỗi S . Thông báo kết quả ra màn hình.

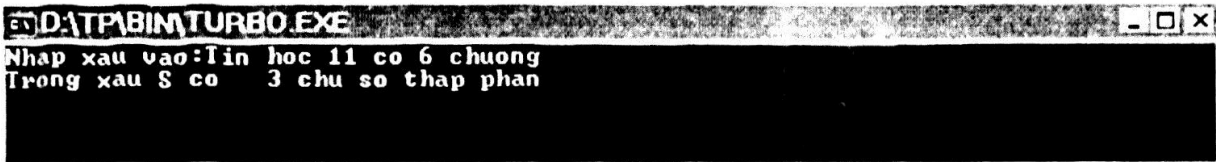
Chương trình:

```

program bai_10_chuong_4;
uses crt;
var
  s: string[100];
  i,dem: integer;
Begin
  clrscr;
  write('Nhap xau vao:');
  readln(s);
  dem:= 0;
  for i:= 1 to length(s) do
    if ('0'<=s[i]) and (s[i]<='9') then dem:=dem+1;
  Writeln('Trong xau S co ',dem,' chu so thap phan');
  readln;
End.
  
```

Khi chạy chương trình, giả sử nhập vào chuỗi $S = \text{'Tin học 11 có 6 chương'}$ thì kết quả chương trình in ra màn hình: $\text{'Trong chuỗi S có 3 chu số thập phân'}$.

Kết quả chương trình có dạng như hình 70 dưới đây:



Hình 70. Kết quả chương trình đếm số thập phân của chuỗi

11. Chương trình xep_loai:

```
program xep_loai;
uses crt;
const max = 60;
type Hocsinh = record
    hoten: string[30];
    ngaysinh: string[10];
    Diachi: string[50];
    Toan, Van: real;
    Xeploai: char;
end;
var
    Lop: array[1..max] of hocsinh;
    N, i: byte;
Begin
    clrscr;
    write('Số lượng học sinh trong lớp N='); readln(N);
    for i := 1 to N do
        begin
            writeln('Nhập số liệu về học sinh thứ', i, ': ');
            write('Họ và tên: '); readln(Lop[i].hoten);
            write('Ngày sinh: '); readln(Lop[i].ngaysinh);
            write('Địa chỉ : '); readln(Lop[i].Diachi);
            write('Điểm Toán: '); readln(Lop[i].Toan);
            write('Điểm Văn : '); readln(Lop[i].Van);
            if Lop[i].Toan + Lop[i].Van >= 18
                then Lop[i].xeploai := 'A';
            if (Lop[i].Toan + Lop[i].Van >= 14) and
                (Lop[i].Toan + Lop[i].Van < 18)
                then Lop[i].Xeploai := 'B';
            if (Lop[i].Toan + Lop[i].Van >= 10) and
                (Lop[i].Toan + Lop[i].Van < 14)
                then Lop[i].Xeploai := 'C';
            if (Lop[i].Toan + Lop[i].Van <= 10)
                then Lop[i].xeploai := 'D';
```



```

        end;
    clrscr;
    writeln('Danh sach xep loai hoc sinh trong lop: ');
    for i:= 1 to N do
writeln(Lop[i].Hoten:30,'- Xep loai:', Lop[i].Xeploai);
        readln
    End.

```

nhằm mục đích xếp loại (*A, B, C, D*) cho học sinh. Để chương trình đưa ra danh sách học sinh xếp loại *A*, thì chúng ta cần đưa thêm đoạn chương trình sau vào cuối chương trình trên:

```

    writeln('Danh sach hoc sinh xep loai A trong lop: ');
    for I:=1 to N do
        if Lop[i].Xeploai ='A' then
            writeln(Lop[i].Hoten:30);

```

Vì vậy, chương trình đưa ra danh sách học sinh xếp loại *A* trong lớp như sau:

```

program xep_loai;
uses crt;
const max= 60;
type Hocsinh = record
    hoten: string[30];
    ngaysinh: string[10];
    Diachi: string[50];
    Toan, Van: real;
    Xeploai: char;
end;
var
    Lop: array[1..max] of hocsinh;
    N,i: byte;
Begin
    clrscr;
    write('So luong hoc sinh trong lop N='); readln(N);
    for i:= 1 to N do
        begin
            writeln('Nhap so lieu ve hoc sinh thu ',i,' : ');
            write('Ho va ten: '); readln(Lop[i].hoten);
            write('Ngay sinh: '); readln(Lop[i].ngaysinh);
            write('Dia chi : '); readln(Lop[i].Diachi);
            write('Diem Toan: '); readln(Lop[i].Toan);
            write('Diem Van : '); readln(Lop[i].Van);
            if Lop[i].Toan+Lop[i].Van>=18 then
                Lop[i].xeploai:='A';
            if (Lop[i].Toan+Lop[i].Van>=14) and
                Lop[i].Toan+Lop[i].Van<18) then

```

```

        Lop[i].Xeploai:='B';
    if (Lop[i].Toan+Lop[i].Van>=10) and
        (Lop[i].Toan+Lop[i].Van<14) then
        Lop[i].Xeploai:='C';
    if (Lop[i].Toan+Lop[i].Van<=10)
        then Lop[i].xeploai:='D';
    end;
clrscr;
    writeln('Danh sach xep loai hoc sinh trong lop: ');
    for i:= 1 to n do
writeln(Lop[i].Hoten:30,' -Xep loai:', Lop[i].Xeploai);
    writeln('Danh sach hoc sinh xep loai A trong lop: ');
    for i:= 1 to N do
        if Lop[i].Xeploai = 'A' then
            writeln(Lop[i].Hoten:30);
    readln
End.

```

Giả sử ta nhập vào dữ liệu của 5 học sinh thì kết quả của chương trình hiện trên màn hình như sau:

```

D:\TP\BIN\TURBO.EXE
Danh sach xep loai hoc sinh trong lop:
    Tran Tuan Anh - Xep loai: A
    Tran Doan Hien - Xep loai: C
    Phan Tuan Hung - Xep loai: B
    Hoang Van Phuong - Xep loai: A
    Nguyen Quyet - Xep loai: A
Danh sach hoc sinh xep loai A trong lop:
    Tran Tuan Anh
    Hoang Van Phuong
    Nguyen Quyet

```

Hình 71. Kết quả chương trình đưa ra danh sách học sinh xếp loại A

II. CÂU HỎI VÀ BÀI TẬP NÂNG CAO

1. Hãy nêu sự giống nhau và khác nhau của mảng một chiều và mảng hai chiều?
2. Xâu là gì? Biến kiểu xâu có thể khai báo như thế nào?
3. Dữ liệu kiểu bản ghi thường được dùng để làm gì? Hãy nêu cách định nghĩa nó.
4. Có mấy cách để gán giá trị cho biến bản ghi? Đó là những cách nào?
5. Viết chương trình nhập vào từ bàn phím số nguyên dương $N(N \leq 100)$ và lấy A gồm N số nguyên $A_1, A_2, \dots, A_N$ có giá trị tuyệt đối không lớn hơn 100. Hãy đưa ra những thông tin sau:
 - Số lượng các giá trị khác nhau trong dãy;
 - Số lượng số nguyên tố trong dãy;

- Số lượng số nguyên tố khác nhau trong dãy;
 - Độ dài lớn nhất của dãy con gồm các số hạng liên tiếp của dãy lập thành cấp số cộng.
6. Cho mảng một chiều 15 phần tử là các số thực. Hãy sắp xếp các số đó theo thứ tự tăng dần.
 6. Một lớp học tập có 4 tổ học tập. Bài kiểm tra được điểm từ 3 đến 10. Ta kí hiệu $A[i, j]$ là số học sinh ở tổ i đạt điểm j . Hãy lập chương trình cho phép:
 - a) Nhập các giá trị $A[i, j]$ từ bàn phím;
 - b) Tính điểm trung bình của bài kiểm tra của lớp đó.

CHƯƠNG V

TỆP VÀ THAO TÁC VỚI TỆP

§14. KIỂU DỮ LIỆU TỆP

A. TÓM TẮT LÝ THUYẾT

1. Vai trò kiểu tệp

Một số đặc điểm của dữ liệu kiểu tệp:

- Dữ liệu kiểu tệp được lưu trữ lâu dài ở bộ nhớ ngoài (đĩa từ, CD, USB,...) và không bị mất khi tắt nguồn điện;
- Lượng dữ liệu lưu trữ trên tệp có thể rất lớn và chỉ phụ thuộc vào dung lượng đĩa.

2. Phân loại tệp và thao tác với tệp

Xét theo cách tổ chức dữ liệu, tệp có thể phân thành hai loại như sau:

- Tệp văn bản là tệp mà dữ liệu được ghi dưới dạng các kí tự theo mã ASCII. Trong tệp văn bản, dãy kí tự kết thúc bởi kí tự xuống dòng hay kí tự kết thúc tệp tạo thành một dòng, chẳng hạn sách báo, giáo án,...
- Tệp có cấu trúc là tệp mà các thành phần của nó được tổ chức theo một cấu trúc nhất định. Tệp nhị phân là một trường hợp riêng của tệp có cấu trúc, ví dụ dữ liệu ảnh, âm thanh,...

Xét theo cách thức truy cập, có thể phân tệp thành hai loại như sau:

- Tệp truy cập tuần tự cho phép truy cập đến một dữ liệu nào đó trong tệp chỉ bằng cách bắt đầu từ đầu tệp và đi qua lần lượt tất cả các dữ liệu trước nó.
- Tệp truy cập trực tiếp cho phép tham chiếu đến dữ liệu cần truy cập bằng cách xác định trực tiếp vị trí (thường là số hiệu) của dữ liệu đó.

Khác với mảng, số lượng phần tử của tệp không xác định trước.

Hai thao tác cơ bản đối với tệp là ghi dữ liệu vào tệp và đọc dữ liệu từ tệp.
Thao tác đọc/ghi tệp được thực hiện với từng phần tử của tệp.

§15. THAO TÁC VỚI TỆP

A. TÓM TẮT LÝ THUYẾT

1. Khai báo

Khai báo biến tệp văn bản có dạng:

```
var <tên biến tệp>: text;
```

Ví dụ

```
var tep1, tep2: text;
```

Tên biến tệp (hay gọi là biến tệp) phải theo đúng quy cách đặt tên. Khai báo biến tệp để sau đó có thể thực hiện các thao tác với tệp thông qua biến tệp.

2. Thao tác với tệp

a) Gắn tên cho biến tệp

Gắn tên tệp với biến tệp thực chất là tạo một tham chiếu giữa tệp trên đĩa (do tên tệp và đường dẫn tương ứng được hệ điều hành xác định) và biến tệp trong chương trình, làm cho biến tệp trở thành đại diện cho tệp. Biến tệp trở thành đối tượng trực tiếp trong chương trình để nhận các thao tác đối với tệp trên đĩa. Gắn tên của một tệp cho biến tệp theo cú pháp:

```
Assign (<biến tệp>, <tên tệp>);
```

Ví dụ

```
Assign(tep1, 'DULIEU.DAT');
```

Trong đó, *tên tệp* là hằng xâu kí tự hoặc giá trị của một biểu thức kiểu xâu kí tự. Tất cả các phép toán trên biến tệp sẽ tác động tới tệp *tên tệp*. Sau khi gọi thủ tục *Assign* khác thực hiện cũng trên *biến tệp* này (nghĩa là lúc đó biến tệp được chuyển sang gắn kết với tệp khác). Tên tệp có thể gồm những đường dẫn chứa ổ đĩa, danh sách các thư mục liên tiếp cách nhau dấu đường dẫn, cuối cùng là tên tệp:

```
<ổ đĩa>:\<tên thư mục>\<tên thư mục>\...\<tên thư mục>\<tên tệp>
```

Ví dụ

```
Assign(tep2, 'C:\INP.DAT');
```

Độ dài lớn nhất của tên tệp là 79 kí tự. Đặc biệt khi tên tệp là xâu rỗng (độ dài xâu bằng 0) thì biến tệp được gán cho các tệp vào/ra chuẩn. Các tệp vào/ra chuẩn được quy định tương ứng với thiết bị nào là tùy thuộc vào sự bổ sung của mỗi chương trình

đích Pascal, nhưng thường quy định tệp *input* chuẩn là bàn phím, tệp *output* chuẩn là màn hình.

b) Mở tệp

- Thủ tục mở tệp để ghi dữ liệu có dạng:

```
rewrite(<biến tệp>);
```

Trong cú pháp, *biến tệp* cần đã được liên kết với một tệp sau khi dùng *Assign*.

Ví dụ

```
assign(tep3, 'C'\KQ.DAT');  
rewrite(tep3);
```

Khi thực hiện *rewrite(tep3)*, nếu trên thư mục gốc của ổ đĩa C chưa có tệp KQ.DAT, thì tệp sẽ được tạo với nội dung rỗng. Nếu đã có, nội dung cũ sẽ bị xóa để chuẩn bị ghi dữ liệu mới.

- Sử dụng thủ tục *reset* mở tệp văn bản đã tồn tại để đọc dữ liệu

Cú pháp:

```
reset(<biến tệp>);
```

Trong cú pháp, *biến tệp* cần phải là đã được gắn kết với một tệp (dùng *assign*). Nếu tệp này không tồn tại thì thực hiện *reset* sẽ gặp lỗi. Nếu tệp đã mở thì nó sẽ đóng lại rồi sau đó mở lại. Vị trí con trỏ tệp sau lời gọi *reset* là đầu tệp.

Ví dụ

Để đọc dữ liệu từ tệp DL.INP, ta có thể mở tệp bằng các thủ tục:

```
tentep:= 'DL.INP';  
assign(tep1, tentep);  
reset(tep1);
```

hoặc

```
assign (tep1, 'DL.INP');  
reset(tep1);
```

c) Đóng/ghi tệp văn bản

- Cú pháp đọc tệp văn bản:

```
Read(<biến tệp>, <danh sách biến>);
```

Hoặc

```
Readln(<biến tệp>, <danh sách biến>);
```

Trong đó, *danh sách biến* là dãy *tên biến 1, tên biến 2, ... tên biến N*. Các dữ liệu cần đọc trong tệp gán vào danh sách biến phải lần lượt có kiểu tương ứng với kiểu của biến trong danh sách biến. Nếu sai kiểu thì chương trình báo lỗi. Lỗi này thường gặp khi biến có kiểu số, dữ liệu được đọc lại là kiểu xâu.

- *Cú pháp ghi tệp văn bản*

`write(<biến tệp>, <danh sách kết quả>);`

hoặc

`writeln (<biến tệp>, <danh sách kết quả>);`

Trong đó, *danh sách kết quả* là dãy *kết quả 1, kết quả 2, ..., kết quả N*. Các *kết quả i* có thể là biến đơn hoặc biểu thức (số học, quan hệ hoặc logic) hoặc hằng xâu.

Ví dụ

Giả sử trong chương trình có khai báo:

`var tệpA, tệpB: text;`

và tệp *tệpA* được mở để đọc dữ liệu, còn tệp *tệpB* dùng để ghi dữ liệu.

Các thủ tục dùng để đọc dữ liệu từ tệp *tệpA* có thể như sau:

`Read(tệpA, A, B, C);`

Hoặc

`Read(tệpA, A, B, C);`

- Một số hàm chuẩn thường dùng trong khi đọc/ghi tệp văn bản:
 - Hàm `eof(<biến tệp>)` trả về giá trị *true* nếu con trỏ tệp đang chỉ tới cuối tệp.
 - Hàm `eoln(<biến tệp>)` trả về giá trị *true* nếu con trỏ tệp đang chỉ tới cuối dòng.

d) Đóng tệp

Thủ tục đóng tệp:

`Close(<biến tệp>);`

Trong cú pháp, *biến tệp* đã được liên kết với một tệp đang mở do đã dùng *reset*, *rewrite* hoặc *append* (*append* chỉ dùng với tệp văn bản) ở thời điểm trước đó để mở tệp.

Ví dụ

`close(tệp1);`

`close(tệp2);`

Sau khi đóng một tệp vẫn có thể được mở lại. Khi mở lại tệp, nếu vẫn dùng biến tệp cũ thì không cần phải dùng thủ tục *assign* gán lại tên tệp.

§16. MỘT SỐ VÍ DỤ VỚI TẬP

A. TÓM TẮT LÝ THUYẾT

Ví dụ 1

Chương trình đọc các cặp tọa độ từ tập TRAI.TXT, tính và đưa ra màn hình khoảng cách (với độ chính xác hai chữ số sau dấu chấm thập phân) giữa hai trại của mỗi giáo viên chủ nhiệm và trại của thầy hiệu trưởng:

```
program khoang_cach;
var d: real;
    f: text;
    x,y: integer;
begin
  assign(f, 'TRAI.TXT');
  reset(f);
  while not eof(f) do
  begin
    read(f,x,y);
    d:= sqrt(x*x+y*y);
    writeln('Khoang cach:',d:10:2);
  readln
  end;
  close(f);
end.
```

Ví dụ 2

Chương trình đọc dữ liệu từ tập REIST.DAT, tính các điện trở tương đương và ghi kết quả ra tập văn bản REIST.EQU, mỗi dòng ghi năm điện trở tương đương của ba điện trở ở dòng dữ liệu vào tương ứng:

```
program Dientro;
var a: array[1..5] of real;
    R1,R2,R3: real;
    i: integer;
    f1,f2: text;
begin
  assign(f1, 'RESIST.DAT');
  reset(f1);
  assign(f2, 'RESIT.EQU');
  rewrite(f2);
  while not eof(f1) do
  begin
    readln(f1,R1,R2,R3);
    a[1]:=R1*R2*R3/(R1*R2+R1*R3+R2*R3);
    a[2]:=R1*R2/(R1+R2)+R3;
```



```

a[3]:=R1*R3/(R1+R3)+R2;
a[4]:=R2*R3/(R2+R3)+R1;
a[5]:=R1+R2+R3;
for i:= 1 to 5 do write(f2,a[i]:3:3,' ');
writeln(f2),
end;
close(f1); close(f2);
end.

```

TÓM TẮT CHƯƠNG V

- Việc trao đổi dữ liệu với bộ nhớ ngoài được thực hiện thông qua kiểu dữ liệu tệp;
- Để có thể làm việc với tệp cần phải khai báo biến tệp;
- Mỗi ngôn ngữ lập trình đều có các hàm/thủ tục chuẩn để làm việc với tệp;
- Các thao tác với tệp văn bản:
 - Cách khai báo biến tệp, mở tệp và đóng tệp;
 - Đọc/ghi: tương tự như làm việc với bàn phím và màn hình.

B. CÂU HỎI VÀ BÀI TẬP

I. BÀI TẬP CƠ BẢN

1. Nêu một số trường hợp cần phải dùng tệp.
2. Trong sơ đồ thao tác với tệp, khi cần nhập dữ liệu từ tệp phải dùng những thao tác nào?
3. Tại sao phải dùng câu lệnh mở tệp trước khi đọc/ghi tệp?
4. Tại sao phải dùng câu lệnh đóng tệp sau khi đã kết thúc ghi dữ liệu vào tệp?

Hướng dẫn trả lời câu hỏi và bài tập

1. Một số trường hợp cần phải dùng tệp, đó là: lưu trữ lượng thông tin lớn, dùng lâu dài.
2. Trong sơ đồ thao tác với tệp, khi cần nhập dữ liệu từ tệp phải dùng những thao tác như gán tên tệp, mở tệp để ghi, ghi dữ liệu vào tệp, đóng tệp để hoàn tất việc ghi dữ liệu.

```

assign(f, fi);
rewrite(f);
write(f, x, ' ', y, ' ', z); {ghi giá trị các biến x, y, z vào tệp}
close(f);

```

3. Trước khi sử dụng tệp phải có câu lệnh mở tệp để trình dịch biết thực hiện mục đích mở tệp để đọc hay ghi, đồng thời đặt con trỏ tệp vào vị trí thích hợp.
4. Phải dùng câu lệnh đóng tệp sau khi đã kết thúc ghi dữ liệu vào tệp để thông báo việc ghi dữ liệu ra tệp.

II. BÀI TẬP NÂNG CAO

1. Kiểu dữ liệu tệp có những đặc điểm nào?
2. Xét theo phương diện tổ chức dữ liệu hoặc theo phương diện truy cập thì tệp có thể phân thành mấy loại? Đó là những loại nào?
3. Hãy nêu ý nghĩa của các hàm **eof**(<biến tệp>) và **eoln**(<biến tệp>).
4. Hãy nêu các thao tác cơ bản với tệp văn bản.
5. Hãy nêu sơ đồ tổng quát về các thao tác với tệp.

CHƯƠNG VI

CHƯƠNG TRÌNH CON VÀ LẬP TRÌNH CÓ CẤU TRÚC

§17. CHƯƠNG TRÌNH CON VÀ PHÂN LOẠI

A. TÓM TẮT LÝ THUYẾT

1. Khái niệm chương trình con

- ❖ *Chương trình con* là một dãy lệnh mô tả một số thao tác nhất định và có thể được thực hiện (được gọi) từ nhiều vị trí trong chương trình.
- ❖ *Một số lợi ích của việc sử dụng chương trình con:*
 - Để tránh được việc phải viết lặp đi lặp lại cùng một dãy lệnh nào đó, ngôn ngữ lập trình cho phép tổ chức dãy lệnh đó thành một chương trình con. Sau đó, mỗi khi chương trình chính cần đến dãy lệnh này thì chỉ cần gọi thực hiện chương trình con đó.
 - Hỗ trợ việc thực hiện các chương trình lớn: khi phải viết chương trình lớn hàng nghìn, hàng vạn lệnh, cần huy động nhiều người tham gia, có thể giao cho mỗi người (hoặc mỗi nhóm) viết một chương trình con, rồi sau đó lắp ghép chúng lại thành chương trình chính.
 - Phục vụ cho quá trình trừu tượng hóa: Người lập trình có thể sử dụng các kết quả được thực hiện bởi chương trình con mà không cần phải quan tâm đến việc các chương trình con đó được cài đặt như thế nào. Trừu tượng hóa là tư

tường chủ đạo để xây dựng chương trình nói chung và chương trình có cấu trúc nói riêng.

- *Mở rộng khả năng ngôn ngữ:* Các ngôn ngữ lập trình thường cung cấp phương thức đóng gói các chương trình con nhằm cung cấp như một câu lệnh mới (tương tự như các lệnh gọi thực hiện các hàm và thủ tục chuẩn) cho người lập trình sử dụng mà không cần biết mã nguồn của nó như thế nào.
- *Thuận tiện cho phát triển, nâng cấp chương trình:* Do chương trình được tạo thành từ các chương trình con nên chương trình dễ đọc, dễ hiểu, dễ kiểm tra và hiệu chỉnh. Việc nâng cấp, phát triển chương trình con nào đó, thậm chí bổ sung thêm các chương trình con mới nói chung không gây ảnh hưởng đến các chương trình con khác.

2. Phân loại và cấu trúc của chương trình con

a) Phân loại

Chương trình con gồm hai loại:

- *Hàm (function)* là chương trình con thực hiện một số thao tác nào đó và trả về một giá trị qua tên của nó. Ví dụ $\sin(x)$ nhận giá trị thực x và trả về giá trị $\sin x, \dots$
- *Thủ tục (procedure)* là chương trình con thực hiện các thao tác nhất định nhưng không trả về giá trị nào đó qua tên của nó. Ví dụ các thao tác vào/ra chuẩn hay thủ tục xử lý xâu:
`writeln, readln, delete, insert, ..`

b) Cấu trúc chương trình con

<phần đầu>

[<phần khai báo>]

<phần thân>

- **Phần khai báo** có thể có khai báo biến cho dữ liệu vào và ra, các hằng và biến dùng trong chương trình con.
- **Phần thân** của chương trình con là dãy câu lệnh thực hiện để từ những dữ liệu vào ta nhận được dữ liệu ra hay kết quả mong muốn.
 - Các biến được khai báo cho dữ liệu vào/ra được gọi là *tham số hình thức*.
 - Các biến được khai báo để dùng riêng trong chương trình con được gọi là *biến cục bộ*.
 - Các biến được khai báo của chương trình chính được gọi là *biến toàn cục*.

c) Thực hiện chương trình con

- Để thực hiện (gọi) một chương trình con, ta cần phải có lệnh tương tự lệnh gọi hàm hay thủ tục chuẩn, bao gồm tên chương trình con với tham số (nếu có) là các hằng và biến chứa dữ liệu vào và ra tương ứng với các tham số hình thức đặt trong cặp ngoặc (và). Các hằng và biến này được gọi là *tham số thực sự*.
- Khi thực hiện chương trình con, các tham số hình thức để nhập dữ liệu vào sẽ nhận giá trị của tham số thực sự tương ứng, còn các tham số hình thức để lưu trữ dữ liệu ra sẽ trả giá trị đó cho tham số thực sự tương ứng.
- Sau khi chương trình con kết thúc, lệnh tiếp theo lệnh gọi chương trình con sẽ được thực hiện.

§18. VÍ DỤ VỀ CÁCH VIẾT VÀ SỬ DỤNG CHƯƠNG TRÌNH CON

A. TÓM TẮT LÝ THUYẾT

1. Cách viết và sử dụng thủ tục

a) Cấu trúc

Thủ tục có cấu trúc như sau:

```
Procedure <tên thủ tục>[(<danh sách tham số>)];  
[<phần khai báo>]  
begin  
    [<dãy các lệnh>]  
end;
```

Trong đó:

- *Đầu thủ tục* gồm tên dành riêng **procedure**, tiếp theo là tên thủ tục. Danh sách tham số có thể có hoặc không.
- *Phần khai báo* dùng để xác định các hằng, kiểu, biến và cũng có thể xác định các chương trình con khác được sử dụng trong thủ tục.
- *Dãy câu lệnh* được viết giữa cặp tên dành riêng **begin** và **end** tạo thành thân của thủ tục.

Ví dụ

- Đầu của thủ tục có thể được viết như sau:

```
Procedure Ve_Hcn(chdai, chrong: integer);
```
- Thủ tục hoán đổi hai biến:

```
Procedure hoan_doi(var x, y: integer);  
Var TG: integer;  
Begin
```

```

    TG := x;
    x := y;
    y := TG;
End;

```

- Thủ tục hoán đổi của tham số giá trị và tham số biến:

```

Procedure hoan_doi( x: integer; var y: integer);
Var TG: integer;
Begin
    TG := x;
    x := y;
    y := TG;
End;

```

1. Cách viết và sử dụng hàm

- Điểm khác nhau cơ bản giữa thủ tục và hàm là việc thực hiện hàm luôn trả về giá trị kết quả thuộc kiểu xác định và giá trị đó được gán cho tên hàm.

- Hàm có cấu trúc tương tự như thủ tục, tuy nhiên có khác nhau phần đầu. Phần đầu khai báo một hàm:

Function <tên hàm>[(<danh sách tham số>)]: <kiểu dữ liệu>;

Trong đó: *kiểu dữ liệu* là kiểu dữ liệu của giá trị mà hàm trả về và chỉ có thể là các kiểu *integer, real, char, boolean, string*.

- Cũng như giống thủ tục, nếu hàm không có tham số hình thức thì không cần danh sách tham số. Trong thân hàm cần có lệnh gán giá trị cho tên hàm:

<tên hàm> := <biểu thức>;

Ví dụ 1

- Hàm tính ước chung lớn nhất (UCLN) của hai số nguyên:

```

Function UCLN(x, y: integer): integer; {bắt đầu hàm UCLN}
var sodu: integer;
Begin
    While x<>y do
        begin
            sodu := x mod y;
            x := y;
            y := sodu;
        end;
    UCLN := x;
End; {hết hàm UCLN}

```

- **Sử dụng hàm**

- Việc sử dụng hàm hoàn toàn tương tự với việc sử dụng các hàm chuẩn, khi viết lệnh gọi gồm hàm và tham số thực sự tương ứng với các tham số hình thức.

- Lệnh gọi hàm có thể tham gia vào biểu thức như một toán hạng và thậm chí là tham số của lời gọi hàm, thủ tục khác, ví dụ:

A: 6*UCLN(Tuso, Mauso)+1;

Ví dụ 2

Hàm tìm số nhỏ nhất trong hai số a và b:

```
Function Min(a, b: real):real;
Begin
  If a<b then Min:= a
  else Min:= b;
end;
```



Bài tập và thực hành 6

1. Mục đích, yêu cầu

- Rèn luyện các thao tác xử lý xâu, kỹ năng tạo hiệu ứng chữ chạy trên màn hình;
- Nâng cao kỹ năng viết, sử dụng chương trình con.

2. Nội dung

a) Trước hết, hãy tìm hiểu việc xây dựng hai thủ tục sau đây:

- Thủ tục *CatDan(s1,s2)* nhận đầu vào là xâu *s1* gồm không quá 79 kí tự, tạo xâu *s2* thu được từ xâu *s1* bằng việc chuyển kí tự đầu tiên của nó xuống vị trí cuối cùng. Ví dụ nếu *s1* = 'abcd' thì *s2* = 'bcda'.

```
Type str79= string[79];
Procedure CatDan(s1:str79; var s2: str79);
Begin
  S2:= copy(s1,2,length(s1)-1)+s1[1];
End;
```

- Thủ tục *CanGiua(s)* nhận đầu vào là xâu *s* gồm không quá 79 kí tự, bổ sung vào đầu *s* một số dấu cách để khi đưa ra màn hình xâu kí tự trong *s* ban đầu được căn giữa dòng gồm 80 kí tự.

```
Procedure CanGiua(var s: str79);
Var i, n: integer;
Begin
  n:= length(s);
  n:= (80-n) div 2;
  for i:= 1 to n do s:= ' ' + s;
end;
```

b) Theo dõi cách sử dụng hai thủ tục trên, ta có thể viết chương trình sau đây để nhập một chuỗi ký tự từ bàn phím và đưa chuỗi đó ra màn hình có dạng dòng chữ chạy giữa màn hình văn bản 25x80.

```
uses crt;
type str79 = string[79];
var s1, s2: str79;
    stop: boolean;
procedure CatDan(s1: str79; var s2: str79);
begin
    s2:= copy(s1,2,length(s1)-1)+s1[1];
end
procedure canGiua(var s: str79);
var i, n: integer;
begin
    n:= length(s);
    n:= (80-n)div 2;
    for i:= 1 to n do s:=' '+s;
end;
begin
clrscr;
write('Nhap xau s1:'); readln(s1);
CanGiua(s1);
clrscr;
while not(stop) do
begin
    gotoxy(1,12); (*chuyen con tro den dau dong 12*)
    write(s1);
    delay(500); (*Dung 500 miligiay*)
    CatDan(s1,s2);
    s1:=s2;
    stop:=keypressed; (*nhan mot phim bat ky de ket thuc*)
end;
readln
End.
```

Hãy chạy thử chương trình trên với dòng chữ

"...Mung nghìn nam Thang Long – Ha Noi!..."

c) Hãy viết thủ tục *Chuchay(s, dong)* nhận đầu vào là chuỗi *s* gồm không quá 79 ký tự và biến nguyên *dong*, đưa ra chuỗi *s* có dạng chữ chạy ở dòng *dong*. Viết và chạy chương trình có sử dụng thủ tục này.

Hướng dẫn làm bài tập và thực hành 6

1. Mục đích, yêu cầu

- Rèn luyện các thao tác xử lý xâu, kỹ năng tạo hiệu ứng chữ chạy trên màn hình;
- Nâng cao kỹ năng viết, sử dụng chương trình con.

2. Nội dung

a) Hai thủ tục *CatDan(s1,s2)* và *CanGiua(s)* sẽ được dùng trong một chương trình để làm một dòng chữ chạy trên màn hình.

➤ Thủ tục *CatDan(s1,s2)* tạo nên xâu *s2* từ xâu *s1* nhận đầu vào, sao cho *s2* chính là trạng thái tiếp theo nếu hình dung *s1* dịch sang trái một vị trí trong chuyển dịch vòng. Ta chỉ cần khai báo *s1* là tham số giá trị, nhưng *s2* phải khai báo là tham số biến.

➤ Thủ tục *CanGiua(s)* thêm một số dấu cách ở đầu một xâu *s* sao cho khi đưa ra màn hình dòng chữ của xâu *s* nằm giữa màn hình. Nếu khai báo *s* là tham số biến thì thủ tục này không có hiệu lực gì vì lệnh đưa *s* ra màn hình không nằm trong thủ tục này.

b) Chương trình làm một dòng chữ chạy trên màn hình. Nó sử dụng hai thủ tục mà chúng ta đã tìm hiểu ở câu a).

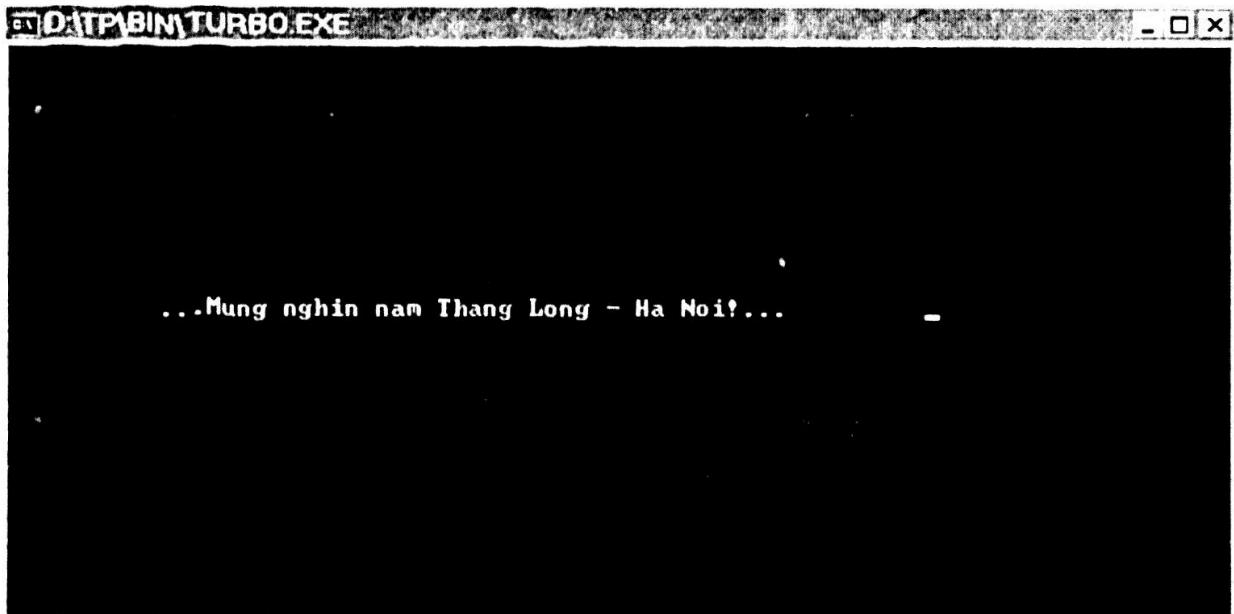
Trong chương trình:

- Thủ tục chuẩn *gotoxy(x,y)* chuyển con trỏ màn hình đến vị trí cột *x* dòng *y* trên màn hình;
- Thủ tục *delay(n)* dừng trạng thái của màn hình trong *n* miligiây.
- Hàm chuẩn *keypressed* không có tham số trả ra giá trị *true* khi có một phím được gõ.

Khi chạy chương trình trên với dòng chữ

'...Mung nghìn nam Thang Long – Ha Noi!...'

Kết quả chương trình có dạng như hình 72 dưới đây:



Hình 72. Kết quả chương trình

c) Để giải quyết bài toán tổng quát hơn (xâu chữ chạy ở dòng bất kỳ do chương trình chính quy định). *Chúng ta cần lưu ý một số điểm sau:*

- Nhiệm vụ của thủ tục *Chuchay(s, dong)* thực hiện, về cơ bản là nhiệm vụ mà chương trình ở câu b) đã làm, chỉ khác là chương trình ở câu b) luôn cho dòng chữ chạy ở dòng 12, còn thủ tục *Chuchay(s, dong)* có tham số *dong* quy định dòng nào trên màn hình xảy ra chuyển động của chữ. Từ ý nghĩa sử dụng tham số *dong* ta thấy chỉ cần khai báo nó là tham số giá trị. Như vậy, thủ tục *Chuchay(s, dong)* chỉ viết khác thân chương trình ở câu b) vài chỗ. Chẳng hạn:

```
procedure Chuchay(s1:str79; dong:byte);
var s2: str79;
    stop:boolean;
begin
    clrscr;
    cangiu(s1);
    clrscr;
    stop:=false;
    while not(stop) do
        begin
            gotoxy(1,dong);
            write(s1);
            delay(100); (*Dung 100 miligiay*)
            Catdan(s1,s2);
            s1:=s2;
            stop:=keypressed;
        end;
    end;
```

- Thủ tục *Chuchay(s, dong)* có sử dụng hai thủ tục *Catdan(s1,s2)* và *CanGiua(s)*. Do vậy, chương trình sử dụng thủ tục *ChuChay* vẫn có hai thủ tục đó trong phần khai báo chương trình con, phải đặt phía trên phần khai báo thủ tục *ChuChay*.

- Thân chương trình sử dụng thủ tục *ChuChay* đơn giản vì chỉ cần gọi thủ tục này làm việc. Tất nhiên trước đó cần xác định giá trị của xâu chữ cần chạy (xác định giá trị của biến *s*) và xác định chữ chạy ở dòng nào trên màn hình (xác định tham số thực sự cho tham số *dong* khi gọi thủ tục *ChuChay*). Chương trình chính có thêm biến *dong* thuộc kiểu byte (thực tế số nguyên dương không vượt quá số dòng của màn hình).

Chương trình:

```

Program Xau_chu_chay_o_dong_bat_ky;
uses crt;
type str79 = string[79];
var s1, s2: str79;
    dong: byte;
    stop: boolean;
procedure CatDan(s1: str79; var s2: str79);
begin
    s2:= copy(s1,2,length(s1)-1)+s1[1];
end;
procedure canGiua(var s:str79);
var i, n: integer;
begin
    n:= length(s);
    n:= (80-n)div 2;
    for i:=1 to n do s:=' '+s;
end;
procedure Chuchay(s1:str79; dong:byte);
var s2: str79;
    stop: boolean;
begin
    clrscr;
    cangiuva(s1);
    clrscr;
    stop:= false;
    while not(stop) do
        begin
            gotoxy(1,dong);
            write(s1);
            delay(100); (*Dung 100 miligiay*)
            Catdan(s1,s2);
            s1:=s2;
            stop:=keypressed;
        end;
end;

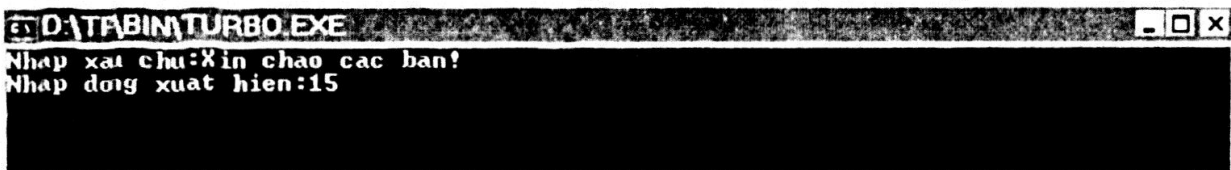
```

```

    end;
  end;
BEGIN
  clrscr;
  write('Nhap xau chu:'); readln(s1);
  write('Nhap dong xuat hien:'); readln(dong);
  chuChay(s1, dong);
  readln
END

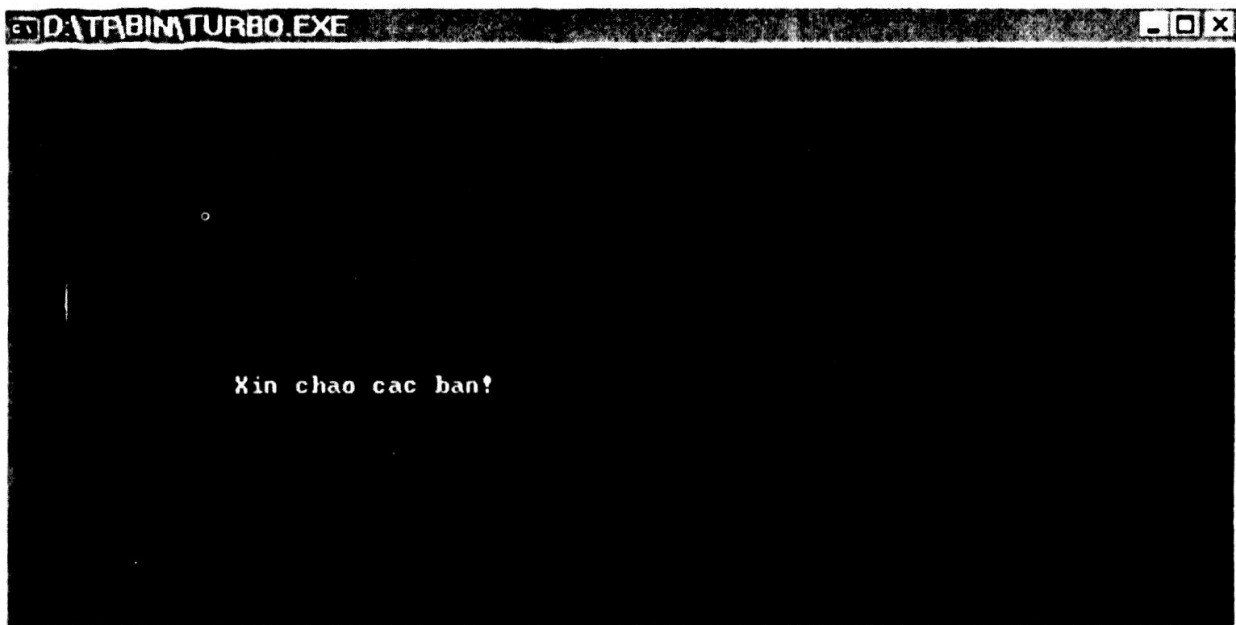
```

Khi chạy chương trình trên, giả sử ta nhập vào xâu:
 'Xin chào các bạn !' thì ngay sau đó sẽ xuất hiện một dòng yêu cầu 'Nhap
 dong xuat hien:' của xâu vừa nhập vào (chẳng hạn, ở dòng 15)(Hình 73).



Hình 73

Kết quả của chương trình sẽ là một dòng chữ chạy ở dòng 15 trên màn hình có dạng
 như hình 74 dưới đây:



Hình 74. Kết quả chương trình xâu chữ chạy ở dòng bất kỳ



Bài tập và thực hành 7

1. Mục đích, yêu cầu

- Nâng cao kỹ năng viết, sử dụng chương trình con.
- Biết cách viết một chương trình có cấu trúc để giải một bài toán trên máy tính.

2. Nội dung

a) Tìm hiểu việc xây dựng các hàm và thủ tục thực hiện tính độ dài các cạnh, chu vi, diện tích, kiểm tra các tính đều, cân, vuông của tam giác được trình bày dưới đây:

Giả thiết tam giác được xác định bởi tọa độ của ba đỉnh. Ta sử dụng kiểu bản ghi để mô tả một tam giác:

type

```
    Diem = record
        x, y: real;
    end;
    Tamgiac= record
        A, B, C:Diem;
    end;
```

Ta xây dựng thủ tục và hàm:

- Thủ tục nhận dữ liệu vào là biến mô tả tam giác R và đầu ra là độ dài của ba cạnh a, b, c:

```
Procedure Daicanh(var R: Tamgiac; var a, b, c:real);
```

- Hàm tính chu vi của tam giác R:
function Chuvi(var R: Tamgiac): real;
- Hàm tính diện tích của tam giác R:
function Dientich(var R:tamgiac): real;
- Thủ tục nhận đầu vào là biến mô tả tam giác R và đầu ra là tính chất của tam giác (*Deu* hay *Can* hay *Vuong*);

```
procedure Tinhchat(var R:Tamgiac;var Deu, Can, Vuong: Boolean);
```

- Thủ tục hiển thị tọa độ ba đỉnh tam giác lên màn hình:
procedure Hienthi(var R: tamgiac);
- Hàm tính khoảng cách giữa hai điểm P, Q:
function Kh_cach(P, Q: Diem): Real;

b) Tìm hiểu chương trình nhập vào tọa độ ba đỉnh một tam giác và sử dụng các hàm, thủ tục được xây dựng dưới đây để khảo sát các tính chất của tam giác.

```

program Tam_giac;
uses crt;
const eps=1.0E-6;
type
    Diem = record
        x, y: real;
    end;
Tamgiac = record
    A, B, C: Diem;
end;
var T: Tamgiac;
    Deu, can, vuong: Boolean;
function Kh_cach(P,Q: Diem): Real;
begin
    Kh_cach:=sqrt((P.x-Q.x)*(P.x-Q.x)+(P.y-Q.y)*(P.y-Q.y));
end;
Procedure Daicanh(var R: Tamgiac; var a, b, c: real);
begin
    a:= Kh_cach(R.B, R.C);
    b:= Kh_cach(R.A, R.C);
    c:= Kh_cach(R.A,R.B);
end;
function Chuvi(var R: Tamgiac): real;
var a, b, c: real;
begin
    Daicanh(R, a, b, c);
    ChuVi:= a+b+c;
end;
function Dientich(var R: tamgiac): real;
var a, b, c, p: real;
begin
    Daicanh(R, a, b, c);
    p:=(a+b+c)/2;
    Dientich:= sqrt(p*(p-a)*(p-b)*(p-c));
end;
procedure Hienthi(var R: tamgiac);
begin
    writeln('Toa do 3 dinh cua tam giac la:');
    writeln('-Dinh A(',R.A.x:0:3,',',R.A.y:0:3,')');
    writeln('-Dinh B(',R.B.x:0:3,',',R.B.y:0:3,')');
    writeln('-Dinh C(',R.C.x:0:3,',',R.C.y:0:3,')');
end;
procedure Tinhchat(var R:Tamgiac; var Deu, Can, Vuong: Boolean);
var a, b, c: real;
begin

```

```

Deu:= false; Can:= false; Vuong:= false;
Daicanh(R, a, b, c);
if (abs(a-b)<eps) and (abs(a-c)<eps) then
  begin
    Deu:= true;
    Can:= true;
  end
else
  if (abs(a-b)<eps) or (abs(a-c)<eps) or (abs(b-c)<eps)
    then Can:= true;
  if (abs(a*a+b*b-c*c)<eps) or (abs(a*a+c*c-b*b)<eps)
    or (abs(b*b+c*c-a*a)<eps) then Vuong:= true;
end;
Begin
  clrscr;
  writeln('Nhap tam giac:');
  write('Toa do dinh A:'); readln(T.A.x, T.A.y);
  write('Toa do dinh B:'); readln(T.B.x, T.B.y);
  write('Toa do dinh C:'); readln(T.C.x, T.C.y);
  writeln('=====');
  Hienthi(T);
  writeln('Dien tich:', dientich(T):9:3);
  writeln('Chu vi:', Chuvi(T):9:3);
  Tinhchat(T, Deu, Can, Vuong);
  writeln('Tam giac co tinh chat:');
  If Deu then writeln('la tam giac deu')
    else if Can then writeln(' la tam giac can');
  if Vuong then writeln('la tam giac vuong');
  readln;
End.

```

c) Viết chương trình sử dụng các hàm và thủ tục xây dựng ở trên để giải bài toán:

Cho tệp dữ liệu TAMGIAC.DAT có cấu trúc như sau:

- Dòng đầu tiên chứa số N ;
- N dòng tiếp theo, mỗi dòng chứa sáu số thực $x_A, y_A, x_B, y_B, x_C, y_C$ là tọa độ ba đỉnh $A(x_A, y_A), B(x_B, y_B), C(x_C, y_C)$ của tam giác ABC.

Hãy nhập dữ liệu từ tệp đã cho và trong số N tam giác đó, đưa ra tệp TAMGIAC.OUT gồm ba dòng:

- Dòng đầu tiên là số lượng tam giác đều;
- Dòng thứ hai là số lượng tam giác cân (nhưng không là đều);
- Dòng thứ ba là số lượng tam giác vuông.

Hướng dẫn làm bài tập và thực hành 7

1. Mục đích, yêu cầu

- Nâng cao kỹ năng viết, sử dụng chương trình con.
- Liệt cách viết một chương trình có cấu trúc để giải một bài toán trên máy tính.

2. Nội dung

a) Câu này nhằm mục đích tìm hiểu việc xây dựng các hàm và thủ tục thực hiện tính độ dài các cạnh, chu vi, diện tích, kiểm tra các tính đều, cân, vuông của tam giác. Cụ thể là:

- Chúng ta cần hiểu rõ cách sử dụng kiểu bản ghi để mô tả một tam giác.
- Xây dựng các thủ tục và hàm đó là:
 - Thủ tục nhận dữ liệu vào là biến mô tả tam giác R và đầu ra là độ dài của ba cạnh a, b, c ;
 - Hàm tính chu vi của tam giác R ;
 - Hàm tính diện tích của tam giác R ;
 - Hàm tính khoảng cách giữa hai điểm P, Q ;
 - Thủ tục hiển thị tọa độ ba đỉnh tam giác lên màn hình;
 - Thủ tục nhận đầu vào là biến mô tả tam giác R và đầu ra là tính chất của tam giác (*Đều* hay *Cân* hay *Vuông*);

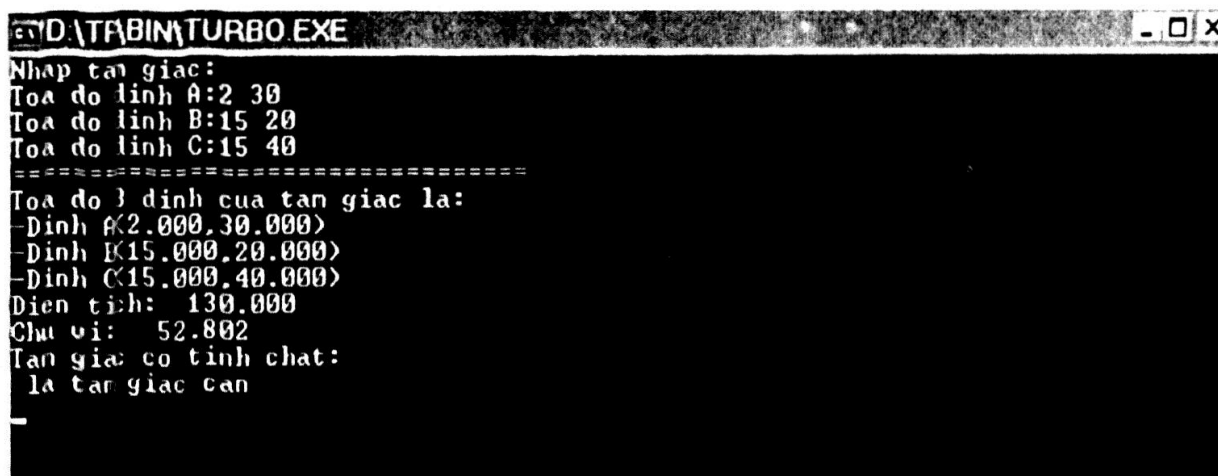
b) Trong câu này các chương trình con giải quyết một số bài toán đơn giản liên quan đến tam giác khi biết tọa độ các đỉnh, đó là: tính độ dài cạnh, tính chu vi, tính diện tích, kiểm tra các tính chất (đều, cân, vuông).

Khi chạy chương trình, giả sử ta nhập tọa độ của các đỉnh A, B, C theo thứ tự là: (2, 30), (15, 20), (15, 40) thì trên màn hình thông báo:

Diện tích: 130.000, Chu vi: 52.802

Tam giác có tính chất: là tam giác cân

Kết quả chương trình như hình 75 dưới đây:



```
D:\T\BIN\TURBO.EXE
Nhap tam giac:
Toa do dinh A:2 30
Toa do dinh B:15 20
Toa do dinh C:15 40
=====
Toa do 3 dinh cua tam giac la:
Dinh A(2.000,30.000)
Dinh B(15.000,20.000)
Dinh C(15.000,40.000)
Diện tích: 130.000
Chu vi: 52.802
Tam giác có tính chất:
là tam giác cân
```

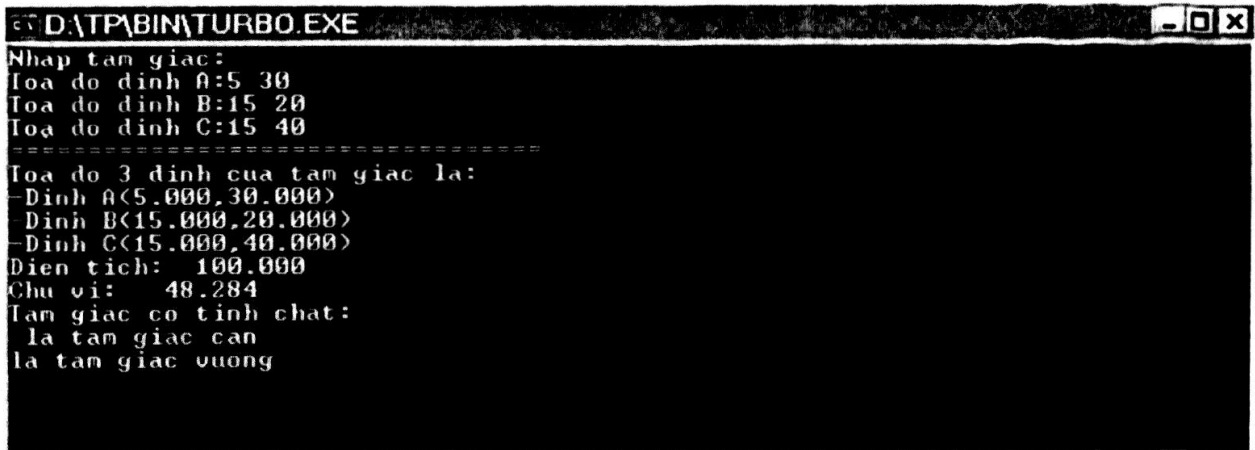
Hình 75

Còn khi nhập tọa độ của các đỉnh A, B, C theo thứ tự là: $(5, 30), (15, 20), (15, 40)$ thì trên màn hình thông báo:

Diện tích: 100.000, Chu vi: 48.284

Tam giác có tính chất: là tam giác vuông cân

Kết quả chương trình như hình 76 dưới đây:



```
D:\TP\BIN\TURBO.EXE
Nhap tam giac:
Toa do dinh A:5 30
Toa do dinh B:15 20
Toa do dinh C:15 40
=====
Toa do 3 dinh cua tam giac la:
-Dinh A(5.000,30.000)
-Dinh B(15.000,20.000)
-Dinh C(15.000,40.000)
Diện tích: 100.000
Chu vi: 48.284
Tam giác có tính chất:
là tam giác cân
là tam giác vuông
```

Hình 76

c) Để xây dựng chương trình có sử dụng các hàm và thủ tục xây dựng ở câu a) trên nhằm đưa ra tính chất các tam giác:

- Số lượng tam giác đều;
- Số lượng tam giác cân;
- Số lượng tam giác vuông.

ta cần tạo tệp tamgiac.dat, để từ tệp này đọc dữ liệu sang tệp của chương trình chính.

- Chương trình tạo tệp tamgiac.dat nhằm mục đích để nhập tọa độ các đỉnh của tam giác.

```
program cau_c_btth7;
uses crt;
type
    Diem = record
        x, y:real;
    end;
    Tamgiac= record
        A, B, C: Diem;
    end;
var i, n: integer;
    f: text;
    T: Tamgiac;
begin
    clrscr;
    assign(f, 'tamgiac.dat');
    rewrite(f);
```

```

write('Nhap so tam giac: '); read(n);
writeln(f,n);
for i:= 1 to n do
    begin
        writeln('Nhap toa cua tam giac ',i,' la:');
        read(T.A.x, T.A.y, T.B.x,T.B.y,T.C.x,T.C.y);
        writeln(f,T.A.x, T.A.y, T.B.x,T.B.y,T.C.x,T.C.y);
    end;
close(f);
readln
end.

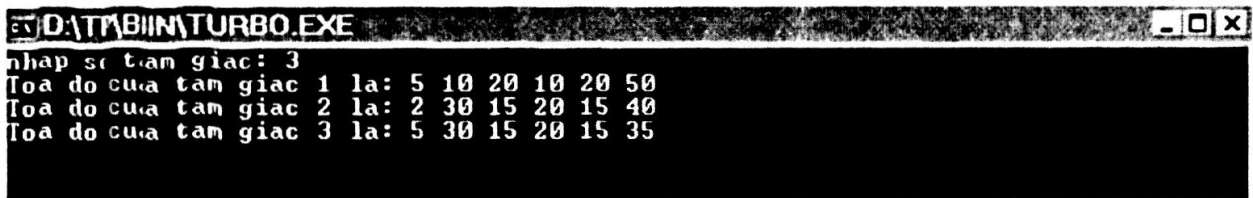
```

Khi chạy chương trình, ta nhập số tam giác $N=3$, với các tọa độ của tam giác lần lượt như sau (Hình 77):

Tọa độ của tam giác 1 là: (5; 10); (20; 10); (20; 50) (đây là tam giác vuông)

Tọa độ của tam giác 2 là: (2;30); (15; 20); (15; 40) (đây là tam giác cân)

Tọa độ của tam giác 3 là: (5; 30); (15; 20); (15; 35) (đây là tam giác thường)



Hình 77

- Chương trình sử dụng các hàm và thủ tục xây dựng ở câu a) trên và có sự trợ giúp của tệp tamgiac.dat như sau:

```

program Tam_giac;
uses crt;
type
    Diem = record
        x, y:real;
    end;
    Tamgiac= record
        A, B, C:Diem;
    end;
const eps=1.0E-6;
var T: Tamgiac;
    D, Cn, V:Boolean;
    N, i, deu, can, vuong: word;
    f: text;
function Kh_cach(P,Q: Diem): Real;
begin
    Kh_cach:=sqrt((P.x-Q.x)*(P.x-Q.x)+(P.y-Q.y)*(P.y-Q.y));
end;
Procedure Daicanh(var R: Tamgiac; var a, b, c:real);

```

```

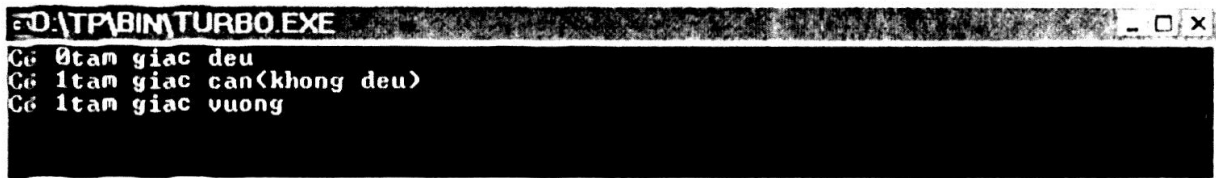
begin
    a:=Kh_cach(R.B, R.C);
    b:=Kh_cach(R.A, R.C);
    c:=Kh_cach(R.A,R.B);
end;
procedure Tinhchat(var R:Tamgiac; var Deu, Can, Vuong: Boolean);
var a, b, c:real;
begin
    Deu:=false; Can:=false; Vuong:=false;
    Daicanh(R, a, b, c);
    if (abs(a-b)<eps) and (abs(a-c)<eps) then
        begin
            Deu:= true;
            Can:= true;
        end
    else
        if (abs(a-b)<eps) or (abs(a-c)<eps) or (abs(b-c)<eps)
            then Can:=true;
        if (abs(a*a+b*b-c*c)<eps) or (abs(a*a+c*c-b*b)<eps)
            or (abs(b*b+c*c-a*a)<eps) then Vuong:= true;
    end;
Begin
    deu:= 0;
    can:= 0;
    vuong:= 0;
    assign(f,'tamgiac.dat');
    reset(f);
    readln(f,N);
    for i:= 1 to N do
        begin
            readln(f, T.A.x, T.A.y, T.B.x,T.B.y,T.C.x,T.C.y);
            Tinhchat(T, D, Cn, V);
            If D then      deu:= deu + 1
                else
                    begin
                        if Cn then can:= can + 1;
                        if V then vuong:= vuong + 1;
                    end;
            end;
        end;
        writeln('Co: ',deu,'tam giac deu');
        writeln('Co: ',can,'tam giac can(khong deu)');
        writeln('Co: ',vuong,'tam giac vuong');
        readln;
    end.

```

Khi chạy chương trình chính này, trên màn hình hiện ra thông báo:

Co 0 tam giác đều
Co 1 tam giác cân(không đều)
Co 1 tam giác vuông

Kt quả chương trình như hình 78 dưới đây:



Hình 78

§19. THƯ VIỆN CHƯƠNG TRÌNH CON CHUẨN

A. TÓM TẮT LÝ THUYẾT

1. CRT

- Thư viện **Crt** chứa các thủ tục liên quan đến việc quản lí và khai thác màn hình, bàn phím của máy tính. Nó dùng để đưa dữ liệu ra màn hình, xây dựng các giao diện màn hình-bàn phím, dùng bàn phím điều khiển chương trình hoặc sử dụng âm thanh để xây dựng các chương trình mô phỏng.
- Thủ tục **Clrscr** dùng để xóa màn hình.
- Thủ tục **TextColor(color)** đặt màu cho chữ trên màn hình, trong đó *color* là hằng hoặc biến xác định màu.
- Thủ tục **TextBackground(color)** đặt màu nền của màn hình, trong đó *color* là hằng hoặc biến xác định màu.
- Thủ tục **GotoXY(x, y)** đưa con trỏ tới vị trí cột *x* dòng *y* của màn hình văn bản.

2. GRAPH

a) Các thiết bị và chương trình hỗ trợ đồ họa

- Màn hình có thể làm việc trong hai chế độ: *chế độ văn bản* và *chế độ đồ họa*.
- Thư viện **Graph** cung cấp các chương trình điều khiển tương ứng với các loại bản mạch đồ họa.
- Độ phân giải màn hình VGA thường được đặt là *640x480*.

b) Khởi tạo chế độ đồ họa

- Các thủ tục, hàm của thư viện **graph** chỉ hoạt động khi chế độ đồ họa đã được thiết lập. Khi kết thúc làm việc với chế độ đồ họa thì cần quay về chế độ văn bản
- Thủ tục thiết lập chế độ đồ họa:

```
Procedure InitGraph(var driver, mode:integer; path:string);
```

Trong đó:

- *driver* là số hiệu của trình điều khiển *BGI*;
- *mode* là số hiệu của độ phân giải;
- *path* là đường dẫn đến các tệp *BGI*.

c) Các thủ tục vẽ điểm, đoạn thẳng

- Thủ tục đặt màu nét vẽ:
`procedure SetColor(color:word);`
- Thủ tục vẽ điểm:
`Procedure PutPixel(x, y:integer;color:word);`

Trong đó:

- *x* và *y* là tọa độ của điểm;
- *color* là màu của điểm.
- Thủ tục vẽ đoạn thẳng nối hai điểm:

`Procedure Line(x1, y1, x2, y2:integer);`

Trong đó: (*x1, y1*) và (*x2, y2*) là các tọa độ của hai điểm đầu, cuối của đoạn thẳng.

- Thủ tục vẽ đoạn thẳng nối điểm hiện tại (vị trí con trỏ) với điểm có tọa độ (*x, y*):

`Procedure LineTo(x, y: integer);`

- Thủ tục vẽ đoạn thẳng nối điểm hiện tại với điểm có tọa độ bằng tọa độ hiện tại cộng với gia số (*dx; dy*):

`Procedure LinRel(dx, dy:integer);`

d) Các thủ tục và hàm liên quan đến vị trí con trỏ

- Các hàm xác định giá trị lớn nhất có thể của tọa độ màn hình *X* và *Y* (để biết độ phân giải màn hình trong chế độ đồ họa đang sử dụng):

`Function GetMaxX: integer;`

`Function GetMaxY: integer;`

- Thủ tục chuyển con trỏ tới tọa độ (*x, y*):

`Procedure MoveTo(x, y:integer);`

e) Một số thủ tục vẽ hình đơn giản

- Vẽ đường tròn có tâm tại (*x, y*), bán kính *r*:

`Procedure Circle(x, y:integer; r:word);`

- Thủ tục vẽ cung của elip có tâm tại điểm (*x, y*) với các bán kính *Xr, Yr* từ góc khởi đầu *StAngle* đến góc kết thúc *EndAngle*:

`Procedure Ellipse(x, y:integer; StAngle, EndAngle,Xr,Yr:word);`

- Thủ tục vẽ hình chữ nhật có các cạnh song song với các trục tọa độ, $(x1; y1)$ là tọa độ của đỉnh trái trên, $(x2; y2)$ là tọa độ của đỉnh phải dưới:

```
Procedure Rectangle(x1, y1, x2, y2:integer);
```

3. Một số thư viện khác

- ❖ **System**: chứa các hàm sơ cấp và các thủ tục vào/ra cho các chương trình;
- ❖ **Dos**: chứa các thủ tục cho phép thực hiện trực tiếp các lệnh như tạo thư mục, thiết lập giờ hệ thống,...
- ❖ **Printer**: Cung cấp các thủ tục làm việc với máy in.

4. Sử dụng thư viện

- ◊ Lệnh khai báo để sử dụng các thủ tục và hàm chuẩn của một số thư viện (trừ *System*):

```
Uses unit1, unit2, ..., unitN;
```

Trong đó: *Uses* là từ khóa; *unit1, unit2, ..., unitN*;

Ví dụ

```
uses crt, dos, graph;
```



Bài tập và thực hành 8

1. Mục đích, yêu cầu

- Giới thiệu một số chương trình để học sinh thấy được khả năng đồ họa của *Pascal*.

2. Nội dung

a) Chương trình sau đây vẽ các đường gấp khúc “ngẫu nhiên” nhờ thủ tục *LineTo*, mỗi đoạn có một màu ngẫu nhiên. Ví trí bắt đầu vẽ là tâm của màn hình. Kết thúc việc vẽ bằng cách nhấn một phím bất kỳ. Chạy thử chương trình và quan sát kết quả trên màn hình.

```
uses crt, graph;
var stop: boolean;
function DetectInit(path:string): integer;
  var drive, mode: integer;
begin
  drive:= 0;
  InitGraph(drive, mode, path);
```



```

        DetectInit:= GraphResult;
    end;
begin
    if DetectInit('C:\TP\BGI')<>0 then
        begin
            write('Loi do hoa! Nhan phim Enter de ket thuc...');
            readln;
        end
    else
        begin
            Randomize;
            MoveTo(getmaxx div 2, Getmaxy div 2);
            stop:= false;
            while not (stop) do
                begin
                    SetColor(Random(GetMaxColor));
                    {Thiet lap mau mot cach ngau nhien}
                    LineTo(Random(getmaxx),Random(getmaxy));
                    Delay(200); {tam dung}
                    stop:=keypressed;
                end;
            end;
            CloseGraph
        End.

```

b) Chương trình dưới đây minh họa việc sử dụng các thủ tục vẽ hình đơn giản. Hãy chạy chương trình rồi thay đổi một số tham số như màu vẽ, tọa độ và quan sát kết quả trên màn hình.

```

program graphDemo;
uses graph;
var
    gd, gm: integer;
    xm, ym, xmaxD4, ymaxD4: word;
begin
    gd:=detect;
    Initgraph(gd, gm, 'C:\TP\BGI');
    xm:=GetmaxX div 2; ym:= GetmaxY div 2);
    {ve hình chu nhật voi net ve mau vang}
    SetColor(yellow);
    Rectangle(10,10, xm, ym);
    readln;
    {Ve duong vong tron mau xanh la cay,tam(450; 100) ban
    kinh 50}
    Setcolor(LightGreen);
    Circle(450, 100, 50);
    readln;
    {ve ellip mau do}

```

```

SetColor (red);
Ellipse(100, 200, 0, 360, 50, 120);
readln;
CloseGraph
end.

```

Hướng dẫn làm bài tập và thực hành 8

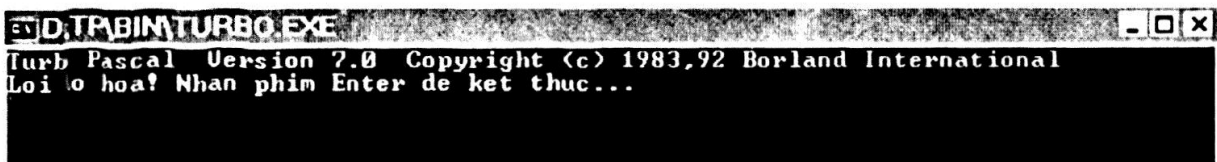
1. Mục đích, yêu cầu

- Giới thiệu một số chương trình để học sinh thấy được khả năng đồ họa của Pascal.

2. Nội dung

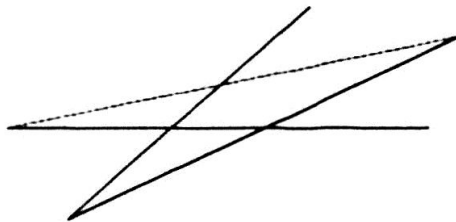
a) Chương trình vẽ các đường gấp khúc “ngẫu nhiên”, mỗi đoạn có một màu ngẫu nhiên.

Trong trường hợp, đường dẫn của chương trình sai (có thể tệp *graph.tpu* không có trên *TP/BGI* hoặc nằm ngoài thư mục *BGI*) thì sẽ có thông báo trên màn hình 'Loi do hoa! Nhan phim Enter de ket thuc...' (Hình 79).



Hình 79

Kh đó, ta nhấn phím *Enter* để quay trở lại chương trình để thay đổi lại đường dẫn. Kết quả chương trình sẽ cho ta các đường gấp khúc “ngẫu nhiên”, mỗi đoạn có một màu ngẫu nhiên, bao gồm các màu xanh, đỏ, tím, vàng,...(Hình 80).



Hình 80

Lưu ý: Nhiều khi tệp *graph.tpu* nằm trong thư mục *BIN* của *TP* của ổ đĩa *D* hoặc *E*,... (© người sử dụng thiết đặt). Đường dẫn đến tệp *graph.tpu* có dạng như sau:

D:\TP\BIN\GRAP.TPU

b) Khi chạy chương trình *GrapDemo*, ta thấy trên màn hình xuất hiện một hình chữ nhật màu vàng. Tiếp đến sau khi gõ phím *Enter*, trên màn hình lại xuất hiện một hình tròn màu xanh. Tương tự, sau khi gõ phím *Enter*, trên màn hình xuất hiện tiếp một hình elip màu đỏ cắt một cạnh của hình vuông.

Khi thay đổi một số tham số như màu vẽ, chẳng hạn hình chữ nhật được thay bằng màu *xanh trời*, tọa độ (20,20,xm,ym),... Đoạn chương trình sau thể hiện sự thay đổi các tham số về màu, tọa độ của hình chữ nhật, hình tròn và hình elip:

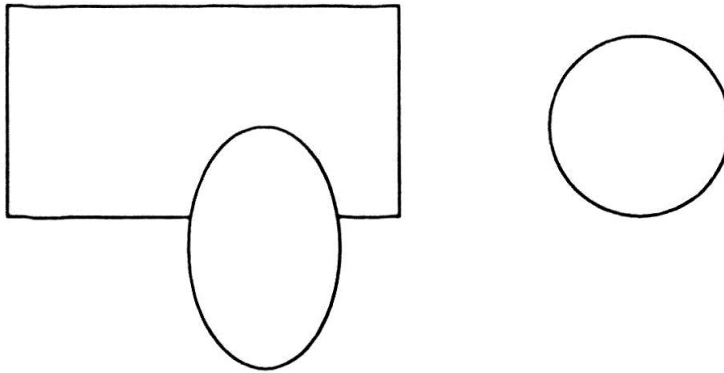
```
xm:= GetmaxX div 2; ym:= GetmaxY div 2;
{ve hình chu nhật voi net ve mau xanh troi}
SetColor(blue);
Rectangle(20,20,xm,ym);
readln;
{Ve duong vong tron mau trang, tam(500; 150) ban kinh
100}
SetColor(red);
Circle(500, 150,100);
readln;
{ve ellip mau tim}
SetColor(magenta);
Ellipse(200, 250, 15, 360, 70, 120);
```

Chương trình:

```
program graphDemo_2;
uses graph;
var
  gd, gm: integer;
  xm, ym, xmaxD4, ymaxD4: word;
Begin
  gd:= detect;
  Initgraph(gd, gm, 'C:\TP\BGI');
  xm:=GetmaxX div 2; ym:= GetmaxY div 2;
  {ve hình chu nhật voi net ve xanh troi}
  SetColor(blue);
  Rectangle(20,20,xm,ym);
  readln;
  {Ve duong vong tron mau do, tam(500; 150) ban kinh 100}
  SetColor(red);
  Circle(500, 150,100);
  readln;
  {ve ellip mau tim}
  SetColor(magenta);
  Ellipse(200, 250, 15, 360, 70, 120);
  readln;
  CloseGraph
End.
```

Khi chạy chương trình vừa đã thay đổi các tham số về tọa độ, màu sắc, trên màn hình xuất hiện một hình *chữ nhật da trời*. Tiếp đến sau khi gõ phím *Enter*, trên màn

hình xuất hiện một *hình tròn màu đỏ*. Tương tự, sau khi gõ phím *Enter*, trên màn hình xuất hiện một *hình Elip màu tím* cắt một cạnh của hình vuông (Hình 75).



Hình 81

TÓM TẮT CHƯƠNG VI

- Chương trình con đóng vai trò quan trọng trong lập trình, đặc biệt trong lập trình có cấu trúc.
- Dùng chương trình con sẽ thuận lợi cho việc tổ chức, viết, kiểm tra chương trình và sử dụng lại.
- Chương trình con có phần đầu, phần khai báo và phần thân.
- Chương trình con có thể có tham số hình thức khi khai báo và được thay bằng tham số thực sự khi gọi. Các tham số hình thức và thực sự phải tương ứng về thứ tự và kiểu dữ liệu.
- Chương trình con được gọi bằng tên của nó;
- Biến được khai báo trong chương trình con là biến cục bộ;
- Thư viện cung cấp những chương trình con chuẩn mở rộng khả năng ứng dụng.

B. CÂU HỎI VÀ BÀI TẬP

I. BÀI TẬP CƠ BẢN

1. Hãy nêu sự giống nhau và khác nhau giữa thủ tục và hàm.
2. Chương trình con có thể không có tham số được không? Cho ví dụ.
3. Hãy cho ví dụ chương trình con có nhiều hơn một kết quả ra.
4. Viết chương trình con (hàm, thủ tục) tính bội số chung nhỏ nhất của hai số nguyên dương a, b . Hãy cho biết trong trường hợp này viết chương trình con dưới dạng hàm hay thủ tục là thuận tiện hơn. Vì sao?

Hướng dẫn trả lời câu hỏi và bài tập

1. Sự giống nhau và khác nhau giữa thủ tục và hàm

- *Giống nhau*: Cả thủ tục và hàm đều là chương trình con, cấu trúc giống như một chương trình trừ dòng đầu tiên và kết thúc bằng END; (thay vì END.). Cả thủ tục và hàm có thể chứa các tham số (tham số giá trị và tham số biến), cùng tuân theo quy định về khai báo và sử dụng các loại tham số này.
- *Khác nhau*: Việc thực hiện hàm luôn trả về giá trị kết quả thuộc kiểu xác định và giá trị đó được gán cho tên hàm.

2. Chương trình con có thể không có tham số. Ví dụ:

```
procedure Ve_Hcn; begin
    writeln('* * * * *');
    writeln('*           *');
    writeln('* * * * *');
end;
```

3. Ví dụ chương trình con có nhiều hơn một kết quả ra:

a) Procedure Hoan_doi(var x, y: integer);

```
var TG: integer;
begin
    TG:= x;
    x:= y;
    y:= TG;
end;
```

b) Procedure Hoan_doi(x: integer; var y: integer);

```
var TG: integer;
begin
    TG:= x;
    x:= y;
    y:= TG;
end;
```

4. Viết chương trình con (hàm, thủ tục) tính bội số chung nhỏ nhất của hai số nguyên dương a, b .

Ta nhận thấy rằng, bội số chung nhỏ nhất của hai số nguyên dương a, b có thể được tính theo công thức:

$$\frac{ab}{d}$$

trong đó d là ước chung lớn nhất của a và b .

Bởi vậy:

Nên viết hàm để tính bội chung nhỏ nhất của hai số nguyên dương vì chương trình con cần trả ra một giá trị;

Hàm tính bội chung nhỏ nhất của hai số nguyên dương a, b cần sử dụng hàm tính ước chung lớn nhất của a và b .

❖ Hàm tính ước chung lớn nhất của hai số nguyên dương a, b :

```
function ucln(a, b: integer): integer;
var r: integer;
begin
    while b > 0 do
        begin
            r := a mod b;
            a := b;
            b := r;
        end;
    ucln := a;
end;
```

❖ Hàm tính bội chung nhỏ nhất của hai số nguyên dương a, b :

```
function bcnn(a, b: integer): integer;
begin
    bcnn := a * b div ucln(a, b);
end;
```

Hi đó, chương trình con tính bội số chung nhỏ nhất của hai số nguyên dương a, b như sau:

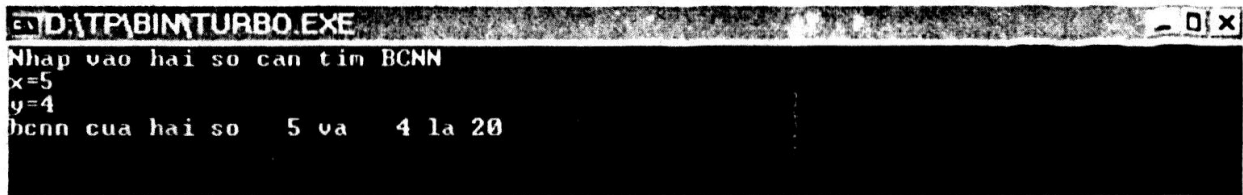
```
program bai4_chuong6;
uses crt;
var
    x, y: integer;
function ucln(a, b: integer): integer;
var r: integer;
begin
    while b > 0 do
        begin
            r := a mod b;
            a := b;
            b := r;
        end;
    ucln := a;
end;
function bcnn(a, b: integer): integer;
begin
```

```

        bcnn:= a*b div ucln(a, b);
    end;
Begin
    clrscr;
    writeln('Nhap vao hai so can tim BCNN');
    write('x='); readln(x);
    write('y='); readln(y);
    writeln('bcnn cua hai so',x:4,'va',y:4,'la',bcnn(a,b) readln;
End.

```

Khi chạy chương trình, giả sử nhập $x = 5$, $y = 4$ thì bội chung nhỏ nhất của x và y là 20. Kết quả chương trình như hình 82 dưới đây:



Hình 82

Như vậy, trong trường hợp này ta viết chương trình con dưới dạng hàm là thuận tiện hơn so với viết chương trình con dưới dạng thủ tục.

II. BÀI TẬP NÂNG CAO

1. Vì sao người ta lại sử dụng chương trình con?
2. Chương trình con có mấy loại? Đó là những loại nào?
3. Hãy nêu những lợi ích khi sử dụng chương trình con?
4. Trong trường hợp nào thì ta viết chương trình con dưới dạng hàm, trong trường hợp nào ta nên viết chương trình con dưới dạng thủ tục?
5. Hãy nêu những ưu thế của việc sử dụng tham biến so với sử dụng tham trị?

MỤC LỤC

LỜI NÓI ĐẦU	3
CHƯƠNG I. MỘT SỐ KHÁI NIỆM VỀ LẬP TRÌNH VÀ NGÔN NGỮ LẬP TRÌNH	5
1. KHÁI NIỆM LẬP TRÌNH VÀ NGÔN NGỮ LẬP TRÌNH	5
2. CÁC THÀNH PHẦN CỦA NGÔN NGỮ LẬP TRÌNH	6
CHƯƠNG II: CHƯƠNG TRÌNH ĐƠN GIẢN	11
3. CẤU TRÚC CHƯƠNG TRÌNH	11
4. MỘT SỐ KIỂU DỮ LIỆU CHUẨN	13
5. KHAI BÁO BIẾN	14
6. PHÉP TOÁN, BIỂU THỨC, CÂU LỆNH GÁN	15
7. CÁC THỦ TỤC CHUẨN VÀO/RA ĐƠN GIẢN.....	18
8. SOẠN THẢO, DỊCH, THỰC HIỆN VÀ HIỆU CHỈNH CHƯƠNG TRÌNH	19
CHƯƠNG III: CẤU TRÚC Rẽ NHÁNH VÀ LẶP	29
9. CẤU TRÚC Rẽ NHÁNH	29
§0. CẤU TRÚC LẶP	30
CHƯƠNG IV: KIỂU DỮ LIỆU CÓ CẤU TRÚC	44
§1. KIỂU MẢNG VÀ BIẾN CÓ CHỈ SỐ.....	44
§2. KIỂU XÂU.....	70
§3. KIỂU BẢN GHI.....	81
CHƯƠNG V: TẬP VÀ THAO TÁC VỚI TẬP	102
§4. KIỂU DỮ LIỆU TẬP	102
§5. THAO TÁC VỚI TẬP.....	103
§6. MỘT SỐ VÍ DỤ VỚI TẬP.....	106
CHƯƠNG VI: CHƯƠNG TRÌNH CON VÀ LẬP TRÌNH CÓ CẤU TRÚC	108
§7. CHƯƠNG TRÌNH CON VÀ PHÂN LOẠI	108
§8. VÍ DỤ VỀ CÁCH VIẾT VÀ SỬ DỤNG CHƯƠNG TRÌNH CON	110
§9. THƯ VIỆN CHƯƠNG TRÌNH CON CHUẨN	125

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI
16 Hàng Chuối - Hai Bà Bà Trưng - Hà Nội
Điện thoại: (04) 39714896; (04) 39724770; Fax: (04) 39714899

Chịu trách nhiệm xuất bản:

Giám đốc: **PHÙNG QUỐC BẢO**
Tổng biên tập: **PHẠM THỊ TRÂM**

Biên tập: **PHAN PHÚC DOÃN**
Sửa bài in: **TRẦN VĂN THẮNG**
Trình bày bìa: **XUÂN VIỆT**

Đối tác liên kết xuất bản:
CÔNG TY SÁCH - TBGD ĐỨC TRÍ

SÁCH LIÊN KẾT

HỌC TỐT TIN HỌC 11

Mã số: 1L - 185ĐH2009

In 3.000 cuốn, khổ 16 x 24cm tại công ty CP IN KHÁNH HỘI

Số xuất bản: 567-2009/CXB/07 - 90/ĐHQGHN, ngày 24/6/2009

Quyết định xuất bản số: 185 LK-TN/XB

In xong và nộp lưu chiểu quý III năm 2009.